Synthesizing Huber's Problem Solving and Kolb's Learning Cycle: A Balanced Approach to Technical Problem Solving

Arnold Kamis
Beverly K. Kahn
Information Systems & Operations Management Department
Sawyer Business School
Suffolk University
8 Ashburton Place
Boston, MA 02108
akamis@suffolk.edu bkahn@suffolk.edu

ABSTRACT

How do we model and improve technical problem solving, such as network subnetting? This paper reports an experimental study that tested several hypotheses derived from Kolb's experiential learning cycle and Huber's problem solving model. As subjects solved a network subnetting problem, they mapped their mental processes according to Huber's problem solving stages by tapping a keypad. Based on Kolb's model, concrete and abstract representations of the subnetting problems were tested to determine whether the form of the problem representation improved performance. For subjects for whom full process data was available, nine of the ten hypotheses were supported. A partial least squares model was developed which explained 27.5 percent of the variance in performance with three predictors. Two of the three predictors for performance were from the Kolb side of the integrated model, whereas the third predictor was from the Huber side. We draw some implications for research and practice, based on the integrated model to explain performance. We conclude that technical problem solving can be modeled as an integration of Kolb's experiential learning cycle and Huber's stages of problem solving. Additional research is needed to extend Kolb's cycle and Huber's stages to other knowledge intensive problem solving domains and to a more diverse set of problem solvers.

Keywords: Technical problem solving, Learning cycle, Process tracing, Partial least squares model

1. INTRODUCTION

One of the most important skills in Internet Protocol (IP) network design and configuration is subnetting, that is, dividing an IP network into smaller networks in a hierarchy, parts of which can be managed separately. Subnetting provides several advantages, such as improved security, ability to manage network resources locally, decreased overall need for unique IP addresses, decreased size of routing tables in core Internet routers, and reduction of broadcast traffic and thus improved utilization of available bandwidth. Without subnetting, it would be impossible to manage organizational IP networks of any significant size. See Kamis and Topi (2007) for a primer on subnetting.

Solving subnetting problems requires an in-depth understanding of IP addressing mechanisms. Once grasped, solving a subnetting problem is not difficult, but for less experienced network designers and administrators, it is often a challenge. As most instructors of networking courses can attest, subnetting and troubleshooting of subnetting problems are among of the most difficult challenges for individuals who are new to networking. (Cigas, 2003; Greca, Cook et al., 2004). A substantial stream of research indicates that intensive laboratory practice is necessary for individuals to solve networking problems in general and subnetting problems in particular (Cigas, 2003; Corbesero, 2003; Greca, Cook et al., 2004). With subnetting, as with programming, one cannot solve problems by reading a book or listening to lectures. Research has shown that an active or problem-based learning approach works better (Roussev and Rousseva, 2004; Whittington, 2004).

To investigate the solving of subnetting problems, we considered two distinct streams of literature: 1) Kolb's Experiential Learning Cycle (Kolb, 1976; Cook and Swain, 1993; Cornwell and Manfredo, 1994; de Ciantis and Kirton, 1996) with the abstract/concrete distinction (Reeves and Weisberg, 1993a; Reeves and Weisberg, 1993b), and 2) problem solving in general (Newell and Simon, 1972) with emphasis on Huber's approach to problem solving (Huber, 1980).

We chose Kolb's cycle because it is a comprehensive and influential model of experiential learning (Kolb, 1976) and because problem solving can be primarily mastered through experiential learning. Kolb's path-breaking work

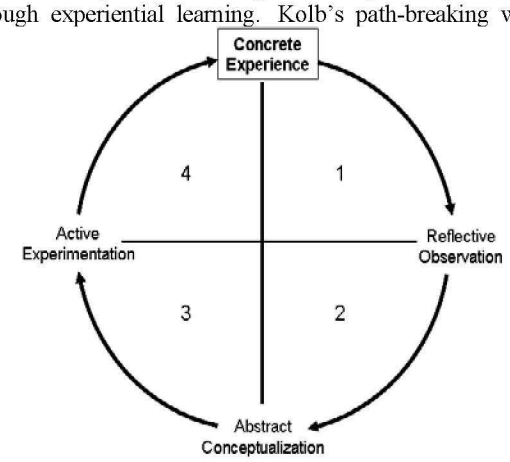


Figure 1: Kolb's Cycle

conceived of experiential learning as a four stage cycle, which, ideally, everyone would master. Within his seminal paper, setting that cycle as the ideal, Kolb claimed that few people could master the complete cycle. Instead, he argued that individuals tend to specialize in one of the four stages or styles of learning. Fortunately, organizations could manage an assortment of individuals, who, collectively, would constitute an organizationally balanced "meta-brain". Rather than follow Kolb's conclusions regarding learning styles, however, we focused on Kolb's distinction between abstract and concrete representations of problems and on an individual's movement between these representations as part of the learning process. Reeves and Weisberg have found that learning by analogy increases with the use of concrete rather than abstract representations (Reeves and Weisberg, 1993a; Reeves and Weisberg, 1993b). Sadoski et al (1993) conducted several experiments that showed that concreteness, i.e., the ease of imagery, had strongly positive impacts on comprehension and recall of information (Sadoski, Goetz et al., 1993).

Our domain of interest was network subnetting, an instance of technical problem solving. Problem solvers learn a set of rules and apply these to solve a network subnetting problem. Problem solving has been studied in linear, well-structured domains, such as chess (Chase and Simon, 1973; Charness, 1992) and in realistic, complex domains, such as the process control industry (Patrick, Gregov et al., 1999). Studies of problem solving can be descriptive (Srinivasan and Te'eni, 1995) or prescriptive (Patrick, Gregov et al., 1999), trace the problem solver's process (Fitzgerald, Simon et al., 2005) or only analyze the outcomes (Kamis and Topi, 2007). The individual problem solver can be viewed as typically reaching adequate solutions, perhaps assisted by technology (Todd and Benbasat, 1992), or inevitably reaching suboptimal solutions, such that computer automation or artificial intelligence is preferred (Garrido and Barber, 2001). Some research starts with actual human problem solving and then strives to help it become more like a theoretical ideal (Steyvers, Tenenbaum et al., 2003).

Our approach in this paper is to triangulate between pure description and normative idealism to bridge the gap between the actual and the ideal. By doing so, we mitigate against the limitations that any one method may have (Covey and Lovie, 1998; Patrick and James, 2004; LeRouge, Garfield et al., 2005; Van Gog, Paas et al., 2005). We hypothesize that a technical subject, such as subnetting, is a good test of Kolb's cycle, and that Kolb's cycle could help us to design better support for the solving of both concrete and abstract problems. In a laboratory setting, we have the problem solver trace his or her problem solving process and then include the tracing in an explanatory structural model of best performance. In technical problem solving, however, we cannot say, a priori, whether concrete problems are less demanding or more demanding than abstract problems. The question has to be answered empirically.

Problem solvers have bounded rationality (Newell and Simon, 1972; Simon, 1972; Simon, 1982). They can pay only so much attention to a problem before they feel overwhelmed. That is why problem solvers make assumptions, partition problems into smaller pieces, strive for adequate if not optimal solutions at first, etc. Simon, for example, defined decision making as Intelligence-Design-Choice (Simon, 1976; Simon, 1977). Intelligence means scouting the decision space, design means developing multiple possible solutions, and choice means selecting the best design. George Huber embedded Simon's decision making stages in a model for managerial decision making: 1) Understand Problem, 2) Plan Solution, 3) Evaluation and Choice of Alternative Solutions, 4) Implementation of Chosen Solution, 5) Monitor and Review the Solution (Huber, 1980). We refer to Huber's multiple-stage model as problem solving. We wondered whether Kolb's learning model could be synthesized with Huber's model of problem solving. In particular, we wondered how the integrated model would explain performance scores.

In sum, we posed the following research questions:

1. Will problem solvers perform better on the abstract or on the concrete version of the same subnetting problem?
2. Will there be any positional effects, i.e., abstract first, concrete second vs. concrete first, abstract second?
3. Which problem solving processes explain performance, i.e., solution correctness, the best?

We first review relevant literature in experiential learning and problem solving. We then develop the hypotheses. We follow with the methods, including a controlled laboratory experiment, and show the results. We then discuss the results and contributions, as well as the limitations, and draw implications for research and practice.

2. PRIOR RESEARCH IN EXPERIENTIAL LEARNING AND PROBLEM SOLVING

According to Kolb's cycle (Kolb, 1976; Moores, Change et al., 2004), people undergo a learning process as described in Figure 1. Learning starts concrete and proceeds to become abstract, but then cycles around to concrete again. The cycle is formed by two axes, concrete experience-abstract conceptualization (i.e., y axis) and reflective observation-active experimentation (i.e., x axis). The two axes form four quadrants and each one is referred to as a stage.

Some researchers, including Kolb, have conceptualized the four quadrants as learning styles. They have found that individuals, professions and organizations may favor a particular learning style (Kolb, 1976; Fatt, 1993; Wu, Dale et al., 1998). For example, individuals taking a Systems Analysis and Design course were found to favor an abstract learning style, the bottom half of the Kolb's cycle (Moores, Change et al., 2004). A fundamental general question remains, however: Should one match the problem solver's preferred style or mismatch it to obtain "brain balance" (Hayes and Allinson, 1996)? This normative ambiguity may explain some of the equivocal results obtained in the information systems training literature (Santhanam and Sein, 1994; Wu, Dale et al., 1998).

Although Kolb's cycle is compelling, the Kolb Learning Style Inventory (KLSI), an instrument based on the theory, has been cited by some researchers as having questionable psychometric properties (Bostrom, Olfman et al., 1990; Moores, Change et al., 2004). In this paper, we do not use the KLSI or the notion of learning styles. Rather, we return to the roots of Kolb's cycle and focus on the individual as the unit of analysis with the aim of supporting every individual. We claim that every individual problem solver performs best by following the entire cycle. Then any teams created from such individuals would be more able to touch all the bases: experiencing, reflecting, thinking and acting.

Kolb's cycle is anchored on the concrete experience of an individual problem solver, whatever the subject being taught. Kolb's work was originally conducted in the context of management education, and it influenced research in other areas as well: nursing education (Fitzpatrick, While et al., 1992), government training (Gray, Hall et al., 1997), organizational learning (Friedman, 2002), technology innovation (Boer and During, 2001; Boer and Gertsen, 2003) and decision support system design (Cook and Swain, 1993). Since subnetting deals with networking hardware configuration, it is clearly an experiential, knowledge-intensive skill. It requires laboratory practice to prove the skill has been learned and to reinforce what the problem solver thinks s/he knows.

Also, consistent with Kolb's cycle is research in computer software training. Computer software training research has shown significant positive impacts of behavior modeling, i.e., computer skill demonstration and hands-on practice, on retention processes (Yi and Davis, 2003). In addition, symbolic mental rehearsal helps improve declarative knowledge and performance by making the problem solver's knowledge structures similar to that of a domain expert (Davis and Yi, 2004). In other words, observing the trainer perform a computer skill, abstracting the skill symbolically, practicing it mentally, and testing one's abstract conceptualization through hands-on practice improves computer skills. Mayer tested the effects of paraphrasing and concrete models on novices learning to program. He found that such techniques did assist students in new situations when administered before the problem (Mayer, 1981). The literature on analogical reasoning shows that analogies are formed between two concrete systems, a source system and a target system (Reeves and Weisberg, 1993b; Reeves and Weisberg, 1994).

Some psychology research shows that problem solving is more concrete than abstract (Reeves and Weisberg, 1993a; Reeves and Weisberg, 1993b; Reeves and Weisberg, 1994). According to Reeves and Weisberg, problem solving is dependent on content and context. Not only are problem solvers unable to form pure abstractions, free of concretes, but they do not even clear their minds of previously solved problems. They solve a current problem by retrieving a prior, concrete problem/solution and forming an analogy to it (Ross and Kennedy, 1990). According to these studies, problem solving by analogy is quite concrete. Accordingly, we should train people starting with concretes in a particular context, and then learn variations in increasingly different content and context.

In sum, Kolb's cycle suggests that learning starts concrete (Kolb, 1976; Reeves and Weisberg, 1993b). Other research suggests that learning can be improved if given appropriate support for problem solving processes (Patrick, Gregov et al., 1999; Bo and Benbasat, 2007). Based on the importance of concrete experience in subnetting, we predicted that the solving of subnetting problems could be modeled with Kolb's cycle. We also theorized that Kolb's model of learning and Huber's model of problem solving could be synthesized.

We posit that Huber's model of problem solving is what takes place within each problem being solved, regardless of whether it is concrete or abstract and regardless of any positioning effects of a problem within a sequence.

3. HYPOTHESES DEVELOPMENT

The same problem can be represented in either a concrete or abstract version. In the concrete version, according to Kolb's cycle, the problem solver may experience cognitive overload and he or she may focus on the wrong aspects of the problem. The concrete version is, in other words, richer and more informative, which can be experienced as engaging and realistic or it can be experienced as overwhelming and challenging to analyze. Alternatively, one can think of the concrete version as a combination of visual and verbal information, whereas the abstract version consists of only verbal information and is therefore less susceptible to focusing incorrectly. (For examples of concrete vs. abstract versions, see Appendix A.) For further evidence that problem solvers focus incorrectly, i.e., on information that is not diagnostic, consider the difficulty that people have in following the prescriptions of Bayes' Rule (Covey and Lovie, 1998).

A concrete version of a problem should be more challenging than an abstract version of the same problem, as illustrated by Kolb's cycle, because the concrete version will have more opportunities for error. Conversely, the abstract problem version has less opportunity for error, because the beginning (stage 1) and the end of the cycle (stage 4) are skipped. There is less switching between stages, less cognitive overload and less opportunity to focus on the wrong aspects of the problem. This may seem counterintuitive, since an abstract problem implies higher order cognitive processes, whereas a concrete problem implies lower order cognitive processes. However, the Kolb cycle implies that the concrete problem is actually a superset

of the abstract problem. The concrete problem traverses the complete cycle, whereas the abstract problem traverses only the bottom half (stages 2 and 3) of the cycle. Our baseline hypothesis is therefore the following:

H1: Problem solvers will perform better on the abstract version of a specific network subnetting problem than they will on the concrete version of the same problem.

We predicted there would be a positioning effect based on concreteness, i.e., that the order of problem versions would matter. Solving a concrete problem first should demand more from the problem solver, setting expectations high for a second, abstract problem. Starting concrete, with the more-demanding problem version, should challenge the problem solver, requiring fuller attention and deployment of his or her mental resources. Success on such a problem would be predictive of success on a subsequent abstract, less demanding problem. Failure on a more-demanding problem would not be predictive of failure on a subsequent, less-demanding problem. Conversely, solving an abstract problem first should demand less from the problem solver, setting expectations low for a subsequent concrete, more-demanding problem. Success on an abstract, less demanding problem version would not be predictive of success on a subsequent concrete, more demanding problem. Failure on an abstract, less-demanding problem would be predictive of failure on a subsequent concrete, more-demanding problem.

H2: Problem solvers given first the concrete version of a specific network subnetting problem will perform better than those given first an abstract version of the same problem.

Concrete representations of problems seem to be more accessible, more visual and more easily graspable than abstract representations. There is a downside, however, to the realism. There is a greater need to figure out where to focus, and to decide which content is relevant and which is not, i.e., seductive yet superfluous details. Good problem solvers need to be able to learn from one domain and apply what they have learned to other domains, often through the transfer and application of their knowledge by analogy (Reeves and Weisberg, 1993a; Reeves and Weisberg, 1993b). Other research has found that concreteness plays a key role in comprehensibility, interest and memory of textual information (Sadoski, Goetz et al., 1993). It takes time and effort to separate the signal from the noise.

To trace the process of problem solving in some detail, we adopt the stages of Huber:

1. Understand Problem
2. Plan Solution
3. Evaluation and Choice of Alternative Solutions
4. Implementation of Chosen Solution
5. Monitor and Review the Solution.

We adapt Huber's framework by combining stages 3 and 4, because in this study there is no separation between choice and implementation of a solution. We thus re-label it Execute Solution. We re-label Monitor and Review to be Check Solution Against Plan or simply Checking. We also added a neutral state for subjects who were not clearly in one of the above problem solving states. (See Table 1.)

In this study, we considered the amount of time spent in each stage as a proportion of the overall amount of time spent solving the whole problem. For example, in solving a problem a subject may spend his time as follows: 10% in Understanding, 20% in Planning, 30% in Executing, 30% in Checking and 10% in Neutral. In general, the more time spent on a prior stage, e.g., Understand Problem, the less time will be spent on all subsequent stages. This is necessarily true, but it is not obvious which of the other states will receive proportionally more time and which will receive proportionally less.

	Stages as Labeled on Keypad
Understand Problem	Understand Problem
Plan Solution	Plan Solution
Evaluation and Choice of Alternative Solutions	Execute Solution
Implementation of Chosen Solution	
Monitor & Review the Solution	Check Solution Against Plan
None of the Above	Neutral

Table 3: Huber's Stages of Problem Solving

We start by predicting that, based on the difficulty of focusing on the relevant aspects of the problem, proportionally more time will be required to understand the problem if it is in concrete form rather than abstract form. Concrete information bounds the problems, however, which could make it simpler.

H3: Problem solvers given the concrete version of a specific network subnetting problem will spend a greater proportion of their time on Understand Problem than problem solvers given the abstract version of the same problem.

Once the problem has been understood, it probably does not matter much whether it is concrete or abstract. We argue, intuitively, that the greater the understanding, the less the problem solver needs to engage in subsequent stages. This has been shown in other contexts, such as learning individually versus working in teams of two (Lim, Ward et al., 1997) and in guarding against erroneous individual habits and assumptions (Patrick, Gregov et al., 1999). This line of reasoning is analogous to the key role played by defining requirements in system or software design. The more precisely the requirements of the system are known, the lower the chances of poor design or implementation which follow. Therefore, we emphasize the key impact of Understand Problem by predicting all of the following:

H4: Problem solvers who spend a greater proportion of their time on Understand Problem will spend a lesser proportion of time on Plan Solution.

H5: Problem solvers who spend a greater proportion of their time on Understand Problem will spend a lesser proportion of time on Execute Solution.

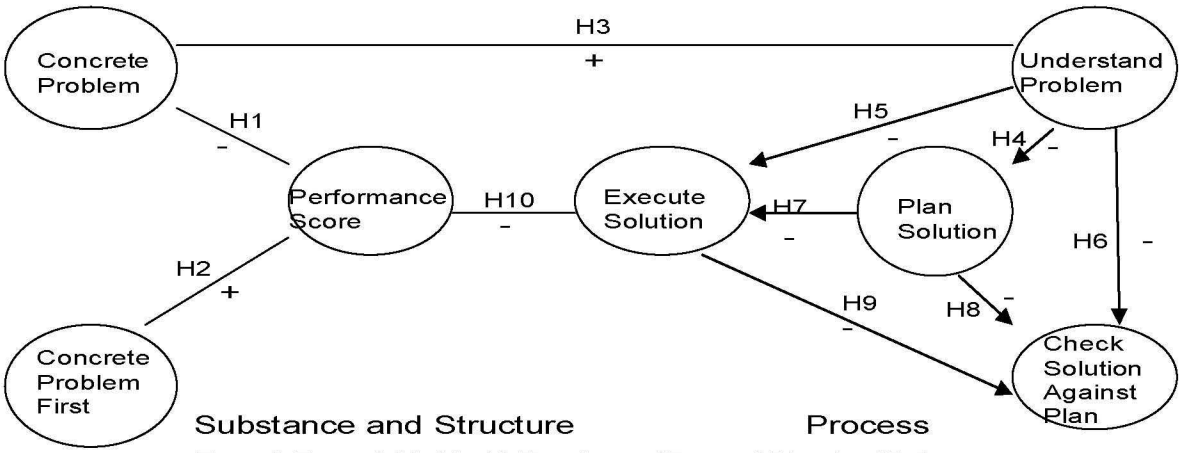


Figure 2: Research Model with Hypotheses of Expected Directional Influence

H6: Problem solvers who spend a greater proportion of their time on Understand Problem will spend a lesser proportion of time on Check Solution Against Plan.

Thus, the more time spent in planning, the less time is needed for Execute Solution, because better planning will focus on more fruitful possibilities and conversely rule out efforts that are "going down the wrong path". Similarly, we expect more planning to reduce the need for Check Solution Against Plan. More planning could lead to a feeling of uncertainty prior to executing the solution, and thus a greater need for checking so the effect will probably be significant but smaller in magnitude than the impact on executing the solution. *H7: Problem solvers who spend a greater proportion of their time on Plan Solution will spend a lesser proportion of time on Execution of Chosen Solution*

H8: Problem solvers who spend a greater proportion of their time on Plan Solution will spend a lesser proportion of time on Check Solution Against Plan.

As the subject spends proportionally more time on executing the chosen solution, the chosen plan will fade increasingly into memory. The plan likely will be considered retrospectively as a non-binding guide, as a point of departure. Conversely, those who spend proportionally less time on executing the chosen plan will be more prone to compare the current execution to the chosen plan.

H9: Problem solvers who spend a greater proportion of their time on Execution of Chosen Solution will spend less time on Check Solution Against Plan.

There are multiple antecedents to Execute Solution, and some of them will be countervailing forces. For example, proportionally greater time in Understand Problem is expected to lead directly to proportionally less time in Execute Solution, but indirectly to proportionally more time in Execute Solution. However, indirect effects are expected to be multiplicative and therefore much smaller in magnitude. Therefore, we rely on the expected impact of Understand Problem to guide our expectation of the Impact Execution Solution on Performance Score. We expect proportionally less time in Execute Solution to lead to greater Performance Score, primarily because of proportionally greater time spent in Understand Problem.

H10: Problem solvers who spend a greater proportion of their time on Execution of Chosen Solution will obtain a lower Performance Score

Figure 2 contains the research model comprising all ten hypotheses.

In sum, the structural hypotheses are tests of problem solving in concrete versus abstract versions and an order effect. The process hypotheses are tests of greater proportion of time spent in an earlier stage of Huber's problem solving model leading to lesser proportion of time spent in all subsequent stages.

4. METHODS

We chose a laboratory experiment to rigorously test the above hypotheses, (i.e., to maximize internal validity). This controlled approach, testing participants in a single laboratory room, enabled us to minimize the influence of confounds, e.g., user mortality, user-user contamination, and problem interference from other sources (Campbell and Stanley, 1963; Campbell and Stanley, 1966). After two pretests, a laboratory experiment was conducted with graduate students who volunteered to participate. The participants were 32 Master of Science in Information

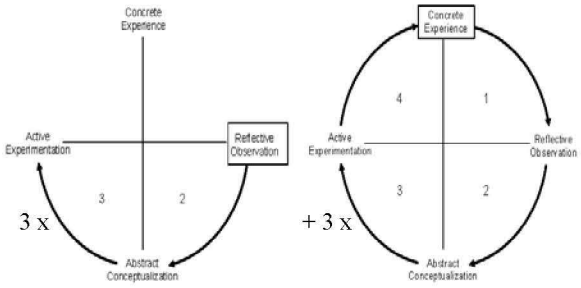


Figure 3: Abstract Problems First

USER			age		yrsCI		yrsNM		Score	
by treatments	Subjects	% male	mean	stdev	mean	stdev	mean	stdev	mean	stdev
Abstract Probs First	15	53%	28.53	2.99	3.80	2.61	0.25	0.77	0.51	0.22
Concrete Probs First	17	71%	25.29	2.51	1.97	1.70	0.15	0.34	0.65	0.22
difference			p<0.05		p<0.05		n.s.		n.s.	
by level of legitimacy	Subjects	% male	mean	stdev	mean	stdev	mean	stdev	mean	stdev
Low Legitimacy	8	63%	26.50	1.41	2.44	1.72	0.09	0.27	0.66	0.24
Medium Legitimacy	10	80%	25.86	2.27	2.16	1.50	0.32	0.93	0.59	0.27
High Legitimacy	14	50%	27.60	4.07	3.49	2.92	0.18	0.36	0.53	0.19
difference			n.s.		n.s.		n.s.		n.s.	

Table 4: Descriptive Statistics at the User level of Analysis

Technology students (37.5% female, 62.5% male) who were homogeneous with regard to age (mean 27, std 3.2). The participants were recruited from a single section of the same graduate level networking course. The total enrollment was 32; all the students volunteered to participate.

4.1. Experimental Design

The concrete and abstract problem versions were designed to have the same level of complexity and difficulty. After completing practice tasks (one concrete and one abstract subnetting problem), half the participants were randomly assigned to solve three abstract problems (also, referred to as the abstract problem set) followed by three concrete problems (see Figure 3).

The other half, randomly sampled, would do three concrete problems (after practice tasks) followed by three abstract problems (see Figure 4).

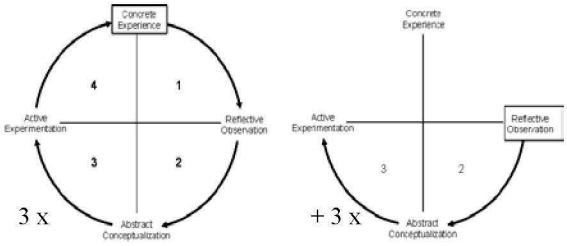


Figure 4: Concrete Problems First

4.2. Experimental procedure

Participants were told that the two top performers (totaling the scores from both problem sets) would receive $100 each. After the lab proctor read the instructions, participants completed the entrance survey to collect demographics and solved two practice problems (10 minutes total). All participants solved one (identical) concrete problem, followed by one (identical) abstract problem. The two problems were of low complexity, designed only to familiarize the participants with the type of problem solving they were about to encounter. Then the participants solved two randomly-assigned problem sets, concrete followed by abstract (or vice versa), each set containing three problems of the same problem type. While solving each problem, the subject was asked to reflect his or her problem solving stage by tapping a key on a keypad (see Appendix B). These stages follow Huber's problem solving model. Table 1 summarizes the stages of problem solving:

Problem solvers were asked to indicate their current problem solving stage by tapping the appropriate key on their keypad. In the analyzed data, the proportion of time spent in each stage was normalized by considering the total proportion of time the problem solving spent solving one problem as 100%. Then, we determined the proportion of time spent in each problem solving stage as a percent of the whole. The time spent solving a problem ranged from 1.5 to 15.6 minutes with a mean of 5 minutes 42 seconds. The hypotheses are based on the proportion of time spent in each problem solving stage.

We chose keypad tapping rather than think-aloud protocol, because vocalizing while problem solving has been shown to interfere with problem solving to some extent, i.e., it is reactive (Lohse and Johnson, 1996; Lim, Ward et al., 1997; Mao and Benbasat, 1998; Patrick and James, 2004). We assumed that keypad tapping would be less reactive because it did not involve vocalization.

Participants were thanked, given an anonymous code for claiming a possible performance prize, and dismissed. All participants who completed the problems obtained 2% extra credit in the class.

5. RESULTS

We analyzed the data to determine the impact of the demographics of our sample. Analysis was conducted on the impact of age, gender, years in the computer industry (i.e., yrsCI), and years in Network Management (i.e, yrsNM). We display the descriptive statistics in Tables 2 and 3. None of these variables had a significant impact on the problem score.

Figure 2 shows some slight differences between the Abstract Problems First (hereafter referred to as Abstract First) and Concrete Problem First (hereafter referred to as Concrete First) treatments. The subjects varied by proportion male, age and years of experience in the computer industry. The Abstract First subjects were older than and had more years of experience in the computer industry than the Concrete First subjects. A superior performance score could be possible in the Abstract First subjects, since their age and experience could be a significant and positive contributing factor. Neither group of subjects indicated a stronger preference or expertise in one type of problem or another. In sum, the two groups of subjects were similar in background

			Score						Time (s)			
PROCESS LEVEL	Instances	% male	mean	stdev		Level of Legitimacy	Instances		mean	stdev		
Abstract Problem	69	64%	0.39	0.46		Medium Legitimacy	57		310.09	173.87		
Concrete Problem	70	63%	0.72	0.31		High Legitimacy	82		341.29	156.26		
difference			p<0.05			difference			n.s.			

USER x PROCESS			age		yrsCl		yrsNM		Score		Time (s)	
within High Legitimacy	Instances	% male	mean	stdev	mean	stdev	mean	stdev	mean	stdev	mean	stdev
Abstract Problem	42	50%	27.64	4.15	3.54	2.95	0.18	0.36	0.70	0.32	320.24	162.46
Concrete Problem	40	50%	27.55	4.04	3.45	2.93	0.19	0.37	0.36	0.45	336.87	165.96
difference			n.s.		n.s.		n.s.		p<0.05		n.s.	
Abstract Problems First	29	38%	31.90	2.09	6.14	2.79	0.00	0.00	0.40	0.41	333.49	154.66
Concrete Problems First	53	57%	25.25	2.74	2.05	1.75	0.28	0.42	0.61	0.41	345.55	158.45
difference			p<0.05		p<0.05		p<0.05		p<0.05		n.s.	

Table 5: Descriptive Statistics at the Process level of Analysis

Average Variance Extracted	score	Concrete Problem	Concrete Problems First	Execute	Verify	Plan	Understand
score	**1.000**						
Concrete Problem	-0.431	**1.000**					
Concrete Problems First	0.235	-0.022	**1.000**				
Execute	-0.178	-0.048	-0.012	**1.000**			
Verify	0.148	-0.092	0.216	-0.174	**1.000**		
Plan	0.118	-0.080	-0.013	-0.401	-0.110	**1.000**	
Understand	-0.008	0.283	-0.181	-0.331	-0.309	-0.105	**1.000**

Table 6: Convergent-Discriminant Validity

and unlikely to be biased. We would have to check, however, whether the Abstract First subjects performed better than the Concrete First subjects.

We examined the keystroke data for each problem of each subject, excluding the practice problems. That gave us detailed process data for six problems per subject. Each problem was examined to see whether it showed a general step pattern, proceeding from one problem solving stage to the next with minimal deviation, an analysis similar to that in Srinivasan and Te'eni (1995). Any problem tracing that did not show this pattern was flagged as less legitimate. Some cases were obvious, such as the keystroke log showing that only one key was pressed the entire time. Some were less obvious, showing the generally expected progression from stage to stage, but perhaps a disproportionately large proportion of time in one stage. This indicated a fixation, for example, because the subject tuned out the keypad and simply tapped one key automatically while turning their attention to the problem solving problem.

The 32 subjects performing 6 problems generated a maximum of 192 possible process traces. Eight subjects clearly did not follow directions. They, the low-legitimacy group, exhibited a range of behaviors, from ignoring the keypad altogether to pressing keys randomly. Omitting their process traces, we were left with 144 medium-to-high legitimacy process traces. Of those 144 process traces, five were deemed to be rare anomalies showing similarity to the behavior of the low-legitimacy process traces and were omitted. Thus, we obtained 139 medium or high legitimacy groups of traces. Table 3 displays descriptive statistics at the process level.

We then examined and compared the medium and high legitimacy groups together and separately. The demographics and scores of the medium and high legitimacy groups were similar. We analyzed the pooled data and found that there was no significant impact from the process variables (Understanding Planning, Execution, Verification, Neutral) on performance score. We noted that the medium legitimacy group was 80% male, whereas the high legitimacy group was 50% male. decided to focus on analyzing only the high legitimacy group, because– in addition to being gender-balanced – they were the ones who took the tasks the most seriously. That left us with 82 high legitimacy traces from 14 subjects. A check of individual performance scores for each subject revealed that they were only moderately correlated. Thus we could not collapse them into an overall performance score or two performance scores, one for the abstract problems and one for the concrete problems.

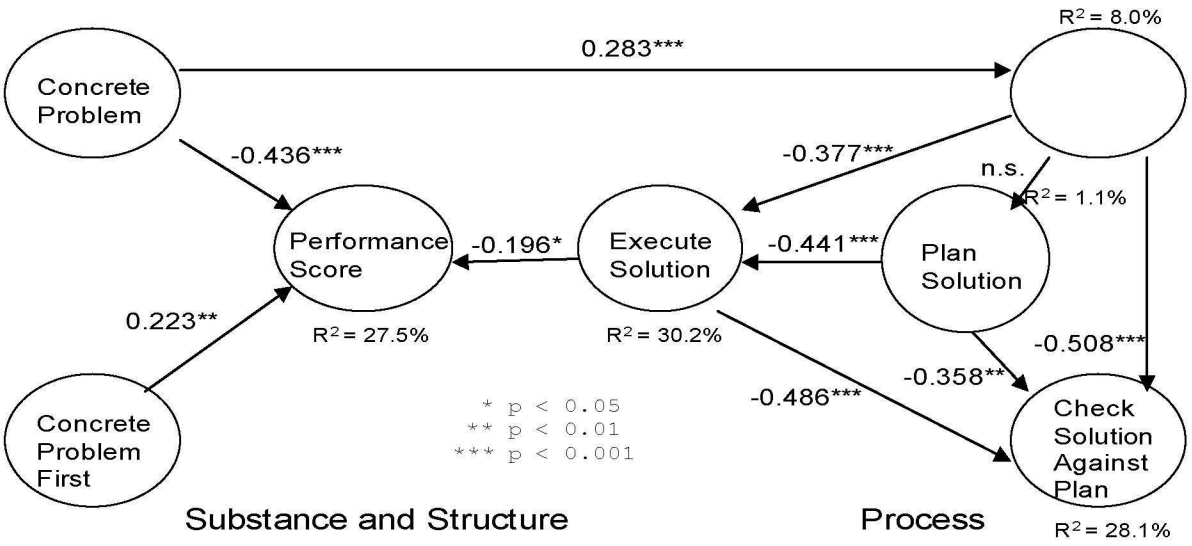


Figure 5: Figure 1: PLS model on High Legitimacy Process Tracings Sample (N=79)

A regression analysis of the treatment variables combined with all the demographic variables showed that only the treatment variables were significant predictors of Performance Score, consistent with Kamis and Topi (2007). That is, all the demographic variables, including age and experience in the computer industry, had negligible effect when the two treatment variables are included. The variance explained was 19.6 percent. We dropped the demographic variables in subsequent analysis with Partial Least Squares (PLS).

We conducted a PLS analysis using PLS-Graph (version 3.0) with the bootstrapping resampling procedure (Chin, 1997; Chin, 1998; Chin, 2000). PLS-Graph is an exploratory form of structural equation modeling that assumes linear relationships between all variables. All constructs were confirmed to have satisfactory convergent and discriminate validity, as show in the cross-loadings of Table 4 (Hulland, 1999; Chin, Marcolin et al., 2003). Each variable's items are more highly correlated internally, within the variable, than they are correlated with the items of other variables. That is, the correlations on the diagonal, the Average Variance Extracted, are larger than those off the diagonal.

A conservative heuristic for sample size in a PLS model is to have at least 10 times the larger of either 1) the largest number of indicators for any construct or 2) the largest number of incoming links to any construct (Chin and Newsted, 1998; Chin, 2000). According to that heuristic, our model requires a minimum sample size of 30. After omitting 3 outliers, our final sample size was 79, well over the minimum sample size.

In sum, legitimate subjects performed considerably better on the abstract problems and experienced an ordering effect. Concrete problems had a significantly lower score. Solving concrete problems before abstract problems helped improve average performance score. High-legitimacy subjects performed considerably better on the abstract problems and experienced an ordering effect. Delving into the problem solving process, proportionally more time spent in non-execution, such as Understand Problem and Plan Solution, led to proportionally less time spent in execution, with 30.2 percent of its variance explained. Proportion of time spent in Execute Solution was a significant factor explaining performance score, yet smaller in magnitude than the abstract versus concrete factor or the ordering effect. Those three predictors explained 27.5 percent of the variance in performance score. All hypotheses except for H4 were supported.

The impact of the proportion of time spent in the first stage, Understand Problem, is analyzed in H4, H5 and H6. As expected, proportionally more time spent in Understand Problem resulted in proportionally less time in the other stages of problem solving, but there was only a significant impact on Execute Solution (H5) and Check Solution Against Plan (H6). Given the lack of positive or negative impact, it appears that individuals neither increase nor decrease their proportion of time in the Plan Solution stage of problem solving based on their proportion of time spent in the Understand Problem stage.

Hypotheses H7, H8 and H9 were substantiated and demonstrate that the proportion of time spent in one stage does negatively impact other stages. H7 and H8 evaluate the impact of the amount spent on Plan Solution on Execute Solution and Check Solution Against Plan respectively. This is also true for the impact of Execute Solution on Check Solution Against Plan.

Only the proportion of time spent in Execution significantly impacted the problem solver's performance score, i.e., the objectively graded score of the subnetting problem. The more time proportionally spent in Execution, the lower the problem solver's score. It could be that individuals who had the most difficulty with the problem spent proportionally more time in the Execution stage. Additionally, these individuals may have been more unsure, less confident in their solution and evaluated more alternative solutions. These individuals often continued

second guessing themselves and usually selected the last but not necessarily the best solution.

6. DISCUSSION

Integrating models from Kolb and Huber, we can better explain technical problem solving. Kolb's model provides information on the representation of the problem (concrete versus abstract), which has significant impacts on the problem solver. In this study, two different treatments were used to describe a problem, concrete and abstract. (Examples of each are in Appendix A.) Scores on the concrete treatment were significantly lower than the scores on the abstract one. Additionally, it was shown that doing the concrete problems first improved overall performance score. Tested individually using multiple regression and PLS, problem version and problem version order were statistically significant at a p-level of 0.05.

Applying Huber's model of problem solving allowed us to better structure the process used to solve a problem. It dealt with the proportion of time spent in each problem solving stage. In the problem solving process, the proportion of time spent in the execution stage had a direct and significant impact on performance score. Spending proportionately less time on execution was helpful, whereas spending proportionally more time on non-execution (understanding, planning and checking), was also helpful for improving performance score.

The results are consistent with Kolb's cycle of experiential learning. In the abstract problem, participants skipped half of the cycle. In the concrete problem, participants had to execute more mental skills and handle more opportunities for error within a concrete representation of the problem. Traversing the full cycle was more demanding; the concrete parts of the cycle challenged the problem solver to focus on and select the relevant aspects of a scenario. Examining the concrete problems for whether they were first or second, we saw a positioning effect on score. Students solving the concrete problems first (before the abstract problems) scored significantly higher than students solving the abstract problems first (before the concrete problems).

Starting with abstractions may be desirable from a theoretical standpoint, but it seems to be hazardous to problem solver performance. If abstract problems are offered as a mental "warm-up," intended to build intellectual scaffolding and confidence, it is counterproductive. It avoids challenging the students to focus, select, and logically induce the relevant aspects of a concrete problem. It sets difficulty expectations lower, and then the subsequent concrete problem is perceived as more challenging. Solving concrete problems first gives the participants the full Kolb cycle, whereas solving abstract problems first gives only the middle part of the Kolb cycle.

The results may be counterintuitive to those who think higher concreteness makes learning deficient versus a more abstract approach. However, the result may well be highly intuitive to those who think hands-on learning (Yi and Davis, 2001) or the case method (Christensen, Garvin et al., 1992) is a better approach to learning versus traditional lecture method. Kolb's cycle takes the problem solver through concrete experience, reflective observation, abstract conceptualization and active experimentation. Undergoing such a process, the student is more likely to be fully challenged and engaged than he or she would with the traditional lecture method.

The argument for learning through cases, i.e., anchoring individuals in authentic situations, is not new (Christensen, Garvin et al., 1992; Zanga, Richard et al., 2004; Carroll and Rosson, 2005). Cases are good for engaging the learner in decisional dilemmas, eliciting empathy toward the decision makers and gaining emotional commitment to solving the problem at hand in an integrative analysis. The results of this study suggest a simple cognitive scaffolding (understand-plan-execute-check) for individuals to bring to bear on an engaging, situated case of technical problem solving. The planning process in particular has been found to be useful in learning through observation of concretes (Zanga, Richard et al., 2004).

7. LIMITATIONS

Several considerations should be kept in mind when interpreting the results of this study. First, the technical domain examined was network subnetting, which may be a unique domain in some respects. The problem solvers used a set of rules learned in the course to solve a subnetting problem. Second, the sample size of thirty-two limited the complexity of the statistical models which could be analyzed with acceptable power. Each student solved six subnetting problems, which resulted ultimately in 79 highly legitimate process traces, following Huber's model to a significant extent. The sample was, on the one hand, a convenience sample, but on the other hand, the enrollment of the sampled networking class was thirty-two, a 100% participation rate. The participants were all graduate students in one MS in Information Technology course being taught by one instructor in one semester. Further research should replicate and extend this study to other technical courses, e.g., programming and database management, including at the undergraduate level, using other instructors, and over multiple semesters. Doing so would increase the sample size and could generalize these findings.

We did not address individual learning styles, i.e., Kolb's KLSI, in this study. Kolb has revised the KLSI several times, and version 3 of the KLSI does show acceptable internal validity and reliability (Kayes, 2005). When assembling learning teams, Kolb's later research calls for a simpler solution than KLSI-tailored learning support; create teams with individuals possessing diverse learning styles, so that they can "touch all the bases" simply by drawing upon the different strengths of the individuals. Kayes, Kayes and Kolb (2005) suggest that touching all the bases is a good idea for both the individuals and the team itself. The results of the current study are consistent with that suggestion.

We avoided a think-aloud protocol for process tracing because it has been found to be reactive, i.e., to interfere (Lohse and Johnson, 1996; Lim, Ward et al., 1997; Mao and Benbasat, 1998; Patrick and James, 2004). Our keypad protocol for process tracing could be reactive as well, since some attention must be diverted from problem solving to awareness of the individual's problem solving stage. A fair

proportion of subjects did not comply with the keypad protocol, making it impossible to systematically analyze their process. It seems that there was a small performance penalty for being in the high-legitimacy group. That is, a close tracing of one's mental state does divert some resources away from the task at hand. Paying more attention to the meta-problem solving level does mean paying less attention to the problem solving level. Thus, our keypad tapping protocol may be reactive for some types of problem solvers. Nonetheless, both Kolb's cycle and Huber's model do combine to explain problem solving performance.

8. IMPLICATIONS FOR FUTURE RESEARCH AND PRACTICE

The results have direct implications for the design of class time and the design of assessments, i.e., quizzes or exams. Start concrete, abstract as needed, and finish concrete. This sequence, consistent with Kolb's experiential learning cycle, is challenging yet engaging. This approach helps individuals to feel grounded in a concrete problem-solving situation, and then it challenges them to focus on the appropriate aspects of the problem, invoke appropriate abstract knowledge and apply that knowledge to the situation at hand. The opposite, starting a class with abstract concepts or starting an exam with true/false questions, seems to be contra-indicated.

The results also have implications for any instructor who needs to delve into the learner's mental processes. We have shown in this study that it is possible to have learners trace their own mental processes through a well-designed keypad. The simple act of vocalizing ones thoughts can interfere with the task at hand, i.e., produce reactivity. By having learners trace their mental processes without vocalization, we succeeded at obtaining process data while avoiding the pitfalls of reactivity found by other researchers (Shaft, 1997; Patrick and James, 2004; Tenopir, Wang et al., 2008). We have also shown that by focusing on the highly legitimate process tracers, one can learn how the learners' mental processes contribute to problem solving performance.

Although it is possible that network subnetting was idiosyncratically responsible for our results, we think it is unlikely. Any technical or knowledge intensive subject requiring experience would be a good candidate. Any science subject requiring problem solving would also be a good candidate. The only subjects where it may not be appropriate would be purely theoretical ones, e.g., theoretical physics.

When demanding work is expected, problem solvers have been found to work harder rather than reduce their effort (Foos, 1992). The results of this study show that Kolb's cycle is a model that applies to technical problem solving. Problems are more demanding when represented in concrete rather than abstract form, and are thus more challenging. In addition, problems in concrete form are better presented first, so that the abstract problems which follow are relatively easier. Thus, although individuals are known to conserve cognitive effort when given the opportunity (Davis, 1989; Todd and Benbasat, 1993), they rise to a challenge when there is no easy way out. This is an encouraging finding with many implications for research and practice. This study integrated the Kolb's cycle with Huber's decision making stages, thus giving a more complete analysis of substance and form. Kolb helped us understand the sequencing of problems, whereas Huber helped us understand the sequencing of stages within a single problem solving effort.

It seems that one should teach subnetting the way many instructors teach programming, i.e., learning-by-doing (Agarwal, Sambamurthy et al., 2000; Yi and Davis, 2001). To the extent that motivation, reflective observation and active experimentation are important, learning-by-doing is recommended. Having problem solvers successfully build upon working solutions, i.e., by increasing complexity, scope of application, robustness to user error, etc., is a sound strategy. It is not the only strategy, of course, but the results of this study are consistent with much of the research in IT training. Train through increasingly complex examples or contexts. Do not start with abstract principles.

Students and other novices have different cognitive processes versus those of experts when programming (Mao and Benbasat, 2000; Hung, 2003; Vainio and Sajaniemi, 2007). Novices experience cognitive overload more easily (Gray, Clair et al., 2007). Experts are better than novices in decomposing complex problems into simpler ones. Rather than take a top-down decomposition approach, however, some researchers advocate working a variety of concrete problems, growing in complexity or decreasing visible details through "fading" (Gray, Clair et al., 2007). The results of this study are consistent with those of Gray et al., adding that the novice should deploy a general framework of understand-plan-execute-check as each concrete problem is encountered.

Extending the scope of this study to a broader spectrum of problem solvers, i.e., undergraduate or graduate students in a variety of majors, would allow greater generalization of our findings. One could also investigate the impact of problem complexity or difficulty within a discipline. In solving problems, the proportion of time in each problem solving stage is variable. The relative proportion of time in each stage impacts performance. If one could determine the idealized proportions of each stage in problem solving, performance could be enhanced. Additionally, an 'idealized' time allocation to Huber's problem solving stages may depend on the characteristics of the problem. This is an important area for future research.

9. CONCLUSION

This study found that Kolb's cycle and Huber's problem solving do apply to the solving of subnetting problems. It explains why concrete problems can be more demanding and error-prone than solving abstract problems. It shows that the individual should be more careful when traversing the entire cycle to maximize performance. If individuals have different learning styles – which were not assessed in this study – we suggest that teaching to the styles is an incomplete strategy. One should also teach to those parts of Kolb's cycle where problem solvers are weak. Other research suggests that one should both "teach to their strengths" and "teach to their weaknesses," i.e., "touch all the bases" in a learning spiral of experiencing, reflecting, thinking and acting (Kayes, Kayes et al., 2005; Armstrong and Mahmud, 2008). Our hunch is the latter, that it would be better to have every problem

solver traverse the entire cycle. Also, if one becomes stuck when solving technical problems, Kolb's cycle serves as an elegant reference for becoming unstuck. The Huber side of the integrated model shows that spending proportionally more time in Understanding, Planning and Checking corresponds with spending proportionally less time in Execution, which leads to greater problem-solver performance.

10. REFERENCES

Agarwal, R., Sambamurthy, V. and Stair, R. M. (2000) "Research Report: The Evolving Relationship between General and Specific Computer Self-Efficacy--an Empirical Assessment." Information Systems Research, Vol. 11, No. 4, pp. 418-430.

Armstrong, S. J. and Mahmud, A. (2008) "Experiential Learning and the Acquisition of Managerial Tacit Knowledge." Academy of Management Learning & Education, Vol. 7, No. 2, pp. 189-208.

Bo, X. and Benbasat, I. (2007) "E-Commerce Product Recommendation Agents: Use, Characteristics, and Impact." MIS Quarterly, Vol. 31, No. 1, pp. 137-209.

Boer, H. and During, W. E. (2001) "Innovation, What Innovation? A Comparison between Product, Process and Organizational Innovation." International Journal of Technology Management, Vol. 22, No. 1-3, pp. 83-107.

Boer, H. and Gertsen, F. (2003) "From Continuous Improvement to Continuous Innovation: A (Retro)(Per)Spective." International Journal of Technology Management, Vol. 26 No. 8, pp. 805-827.

Bostrom, R. P., Olfman, L. and Sein, M. (1990) "The Importance of Learning Style in End-User Training." MIS Quarterly, Vol. 14, No. 1, pp. 100-119.

Campbell, D. and Stanley, J. (1963) Experimental and Quasi-Experimental Designs for Research. Chicago, IL, Rand-McNally.

Campbell, D. T. and Stanley, J. C. (1966) Experimental and Quasi-Experimental Designs for Research, Houghton Mifflin College.

Carroll, J. M. and Rosson, M. B. (2005) "A Case Library for Teaching Usability Engineering: Design Rationale, Development, and Classroom Experience." Journal on Educational Resources in Computing, Vol. 5, No. 1.

Charness, N. (1992) "The Impact of Chess Research on Cognitive Science." Psychological Research, Vol. 54, No. 1, pp. 4-9.

Chase, W. G. and Simon, H. A. (1973) "The Mind's Eye in Chess" in Visual Information Processing. W. G. Chase. New York, Academic Press.

Chin, W. (1998) "Issues and Opinion on Structural Equation Modeling." MIS Quarterly, Vol. 22, No. 1, pp. 7-16.

Chin, W. (2000) "Frequently Asked Questions – Partial Least Squares & PLS-Graph Home Page." Retrieved 04/12/2002, from http://disc-nt.cba.uh.edu/chin/plsfaq/plsfaq.htm.

Chin, W. W. (1997, October 18, 1997) "Overview of the PLS Method." from http://disc-nt.cba.uh.edu/chin/PLSINTRO.HTM.

Chin, W. W., Marcolin, B. L. and Newsted, P. R. (2003) "A Partial Least Squares Latent Variable Modeling Approach for Measuring Interaction Effects: Results from a Monte Carlo Simulation Study and an Electronic-Mail Emotion/Adoption Study." Information Systems Research, Vol. 14, No. 2, pp. 189-217.

Chin, W. W. and Newsted, P. R. (1998) "Structural Equation Modeling Analysis with Small Samples Using Partial Least Squares" in Statistical Strategies for Small-Sample Research. R. H. Hoyle. Thousand Oaks, CA, Sage Publications, Inc.

Christensen, C. R., Garvin, D. A. and Sweet, A. (1992) Education for Judgment: The Artistry of Discussion Leadership. Cambridge, MA, Harvard Business School Press Books.

Cigas, J. (2003) "An Introductory Course in Network Administration." ACM SIGCSE Bulletin, Vol. 35, No. 1, pp. 113 - 116.

Cook, G. J. and Swain, M. R. (1993) "A Computerized Approach to Decision-Process Tracing for Decision-Support System-Design." Decision Sciences, Vol. 24, No. 5, pp. 931-952.

Corbesero, S. G. (2003) "Teaching System and Network Administration in a Small College Environment." Journal of Computing Sciences in Colleges, Vol. 19, No. 2, pp. 155-163.

Cornwell, J. M. and Manfredo, P. A. (1994) "Kolb Learning Style Theory Revisited." Educational and Psychological Measurement, Vol. 54, No. 2, pp. 317-327.

Covey, J. A. and Lovie, A. D. (1998) "Information Selection and Utilization in Hypothesis Testing: A Comparison of Process-Tracing and Structural Analysis Techniques." Organizational Behavior & Human Decision Processes, Vol. 75, No. 1, pp. 56-74.

Davis, F. D. (1989) "Perceived Usefulness, Perceived Ease of Use, and User Acceptance of Information Technology." MIS Quarterly, Vol. 13, No. 3, pp. 319-340.

Davis, F. D. and Yi, M. Y. (2004) "Improving Computer Skill Training: Behavior Modeling, Symbolic Mental Rehearsal, and the Role of Knowledge Structures." Journal of Applied Psychology, Vol. 89, No. 3, pp. 509-523.

de Ciantis, S. M. and Kirton, M. J. (1996) "A Psychometric Reexamination of Kolb's Experiential Learning Cycle Construct: A Separation of Level, Style, and Process." Educational and Psychological Measurement, Vol. 56, No. 5, pp. 809-820.

Fatt, J. P. T. (1993) "Learning Styles in Training: Teaching Learners the Way They Learn." Industrial & Commercial Training, Vol. 25, No. 9, pp. 17-23.

Fitzgerald, S., Simon, B. and Thomas, L. (2005) "Strategies That Students Use to Trace Code: An Analysis Based in Grounded Theory," International Computing Education Research Workshop, Seattle, WA, ACM.

Fitzpatrick, J. M., While, A. E. and Roberts, J. D. (1992) "The Role of the Nurse in High-Quality Patient-Care - a Review of the Literature." Journal of Advanced Nursing, Vol. 17, No. 10, pp. 1210-1219.

Foos, P. W. (1992) "Test Performance as a Function of Expected Form and Difficulty." Journal of Experimental Education, Vol. 60, No. 3, pp. 205-211.

Friedman, V. J. (2002) "The Individual as Agent of Organizational Learning." California Management Review, Vol. 44, No. 2, pp. 70-89.

Garrido, A. and Barber, F. (2001) "Integrating Planning and Scheduling." Applied Artificial Intelligence, Vol. 15, No., pp. 471-491.

Gray, G. R., Hall, M. E., Miller, M. and Shasky, C. (1997) "Training Practices in State Government Agencies." Public Personnel Management, Vol. 26, No. 2, pp. 187-202.

Gray, S., Clair, C. S., James, R. and Mead, J. (2007) "Suggestions for Graduated Exposure to Programming Concepts Using Fading Worked Examples," Third international workshop on Computing education research, Atlanta, Georgia, USA, ACM.

Greca, A. N., Cook, R. P. and Harris, J. K. (2004) "Enhancing Learning in a Data Communication and Networking Course with Laboratory Experiments." Journal of Computing Sciences in Colleges, Vol. 19, No. 3, pp. 79 - 88

Hayes, J. and Allinson, C. W. (1996) "The Implications of Learning Styles for Training and Development: A Discussion of the Matching Hypothesis." British Journal of Management, Vol. 7, No. 1, pp. 63-73.

Huber, G. P. (1980) Managerial Decision Making, Scott Foresman & Co.

Hulland, J. (1999) "Use of Partial Least Squares (Pls) in Strategic Management Research: A Review of Four Recent Studies." Strategic Management Journal, Vol. 20, No. 2, pp. 195-204.

Hung, S.-Y. (2003) "Expert Versus Novice Use of the Executive Support Systems: An Empirical Study." Information & Management, Vol. 40, No. 3, pp. 177.

Kamis, A. and Topi, H. (2007) "Network Subnetting: An Instance of Technical Problem Solving in Kolb's Experiential Learning Cycle," Hawaii International Conference on System Sciences, Big Island, HI.

Kayes, A. B., Kayes, D. C. and Kolb, D. A. (2005) "Experiential Learning in Teams." Simulation & Gaming, Vol. 36, No. 3, pp. 330-354.

Kayes, D. C. (2005) "Internal Validity and Reliability of Kolb's Learning Style Inventory Version 3 (1999)." Journal of Business & Psychology, Vol. 20, No. 2.

Kolb, D. A. (1976) "Management and the Learning Process." California Management Review, Vol. 18, No. 3, pp. 21-31.

LeRouge, C., Garfield, M. and Kamis, A. (2005) "Effective Champion Networks in Interorganizational Telemedicine Programs," American Telemedicine Association, Denver, Colorado.

Lim, K. H., Ward, L. M. and Benbasat, I. (1997) "An Empirical Study of Computer System Learning: Comparison of Co-Discovery and Self-Discovery Methods." Information Systems Research, Vol. 8, No. 3, pp. 254-272.

Lohse, G. L. and Johnson, E. J. (1996) "A Comparison of Two Process Tracing Methods for Choice Tasks." Organizational Behavior & Human Decision Processes, Vol. 68, No. 1, pp. 28-43.

Mao, J.-Y. and Benbasat, I. (1998) "Contextualized Access to Knowledge: Theoretical Perspectives and a Process-Tracing Study." Information Systems Journal, Vol. 8, No. 3, pp. 217-239.

Mao, J.-Y. and Benbasat, I. (2000) "The Use of Explanations in Knowledge-Based Systems: Cognitive Perspective and a Process-Tracing Analysis." Journal of Management Information Systems, Vol. 17, No. 2, pp. 153-179.

Mayer, R. E. (1981) "The Psychology of How Novices Learn Computer Programming." ACM Computing Surveys, Vol. 13, No. 1, pp. 121-141.

Moores, T. T., Change, J. C.-J. and Smith, D. K. (2004) "Learning Style and Performance: A Field Study of Is Students in an Analysis and Design Course." Journal of Computer Information Systems, Vol. 45, No. 1, pp. 77-85.

Newell, A. and Simon, H. A. (1972) Human Problem Solving. Englewood Cliffs, NJ, Prentice Hall.

Patrick, J., Gregov, A., Halliday, P., Handley, J. and O'Reilly, S. (1999) "Analysing Operators' Diagnostic Reasoning During Multiple Events." Ergonomics, Vol. 42, No. 3, pp. 493-515.

Patrick, J. and James, N. (2004) "Process Tracing of Complex Cognitive Work Tasks." Journal of Occupational and Organizational Psychology, Vol. 77, No. 2, pp. 259-280.

Reeves, L. M. and Weisberg, R. W. (1993a) "Abstract Versus Concrete Information as the Basis for Transfer in Problem Solving: Comment on Fong and Nisbett (1991)." Journal of Experimental Psychology, Vol. 122, No. 1, pp. 125-128.

Reeves, L. M. and Weisberg, R. W. (1993b) "On the Concrete Nature of Human Thinking: Content and Context in Analogical Transfer." Educational Psychology, Vol. 13, No. 3/4, pp. 245-258.

Reeves, L. M. and Weisberg, R. W. (1994) "The Role of Content and Abstract Information in Analogical Transfer." Psychological Bulletin, Vol. 115, No. 3, pp. 381-400.

Ross, B. H. and Kennedy, P. T. (1990) "Generalizing from the Use of Earlier Examples in Problem Solving." Journal of Experimental Psychology: Learning, Memory, And Cognition, Vol. 16, No. 1, pp. 42-55.

Roussev, B. and Rousseva, Y. (2004) "Active Learning through Modeling: Introduction to Software Development in the Business Curriculum." Decision Sciences Journal of Innovative Education, Vol. 2, No. 2, pp. 121-152.

Sadoski, M., Goetz, E. T. and Fritz, J. B. (1993) "Impact of Concreteness on Comprehensibility, Interest, and Memory for Text - Implications for Dual Coding Theory and Text Design." Journal of Educational Psychology, Vol. 85, No. 2, pp. 291-304.

Santhanam, R. and Sein, M. K. (1994) "Improving End-User Proficiency: Effects of Conceptual Training and Nature of Interaction." Information Systems Research, Vol. 5, No. 4, pp. 378-399.

Shaft, T. M. (1997) "Responses to Comprehension Questions and Verbal Protocols as Measures of Computer Program Comprehension Processes." Behaviour and Information Technology, Vol. 16, No. 6, pp. 320-336.

Simon, H. (1976) Administrative Behavior. New York, The Free Press.

Simon, H. A. (1972) "Theories of Bounded Rationality" in Decision and Organisation. Radner.

Simon, H. A. (1977) The New Science of Management Decision. Upper Saddle River, NJ, Prentice Hall PTR.

Simon, H. A. (1982) Models of Bounded Rationality, MIT Press.

Srinivasan, A. and Te'eni, D. (1995) "Modeling as Constrained Problem Solving: An Empirical Study of the Data Modeling Process." Management Science, Vol. 41, No. 3, pp. 419-435.

Steyvers, M., Tenenbaum, J. B., Wagenmakers, E.-J. and Blum, B. (2003) "Inferring Causal Networks from Observations and Interventions." Cognitive Science, Vol. 27, No. 3, pp. 453-489.

Tenopir, C., Wang, P., Zhang, Y., Simmons, B. and Pollard, R. (2008) "Academic Users' Interactions with Sciencedirect in Search Tasks: Affective and Cognitive Behaviors." Information Processing & Management, Vol. 44, No. 1, pp. 105-121.

Todd, P. and Benbasat, I. (1992) "The Use of Information in Decision Making: An Experimental Investigation of the Impact of Computer-Based Decision Aids." MIS Quarterly, Vol. 16, No. 3, pp. 373-393.

Todd, P. and Benbasat, I. (1993) "An Experimental Investigation of the Relationship between Decision Makers, Decision Aids and Decision Making Effort." INFOR, Vol. 31, No. 2, pp. 80-100.

Vainio, V. and Sajaniemi, J. (2007) "Factors in Novice Programmers' Poor Tracing Skills," Proceedings of the 12th annual SIGCSE conference on Innovation and technology in computer science education, Dundee, Scotland ACM.

Van Gog, T., Paas, F., Van Merriënboer, J. G. and Witte, P. (2005) "Uncovering the Problem-Solving Process: Cued Retrospective Reporting Versus Concurrent and Retrospective Reporting." Journal of Experimental Psychology / Applied, Vol. 11, No. 4, pp. 237-244.

Whittington, K. J. (2004) "Infusing Active Learning into Introductory Programming Courses." Journal of Computing Sciences in Colleges, Vol. 19, No. 5, pp. 249-259

Wu, C.-C., Dale, N. B. and Bethel, L. J. (1998) "Conceptual Models and Cognitive Learning Styles in Teaching Recursion." ACM SIGCSE Bulletin, Vol. 30, No. 1, pp. 292-296.

Yi, M. Y. and Davis, F. D. (2001) "Improving Computer Training Effectiveness for Decision Technologies: Behavior Modeling and Retention Enhancement." Decision Sciences, Vol. 32, No. 3, pp. 521-544.

Yi, M. Y. and Davis, F. D. (2003) "Developing and Validating an Observational Learning Model of Computer Software Training and Skill Acquisition." Information Systems Research, Vol. 14, No. 2, pp. 146-169.

Zanga, A., Richard, J.-F. and Tijus, C. (2004) "Implicit Learning in Rule Induction and Problem Solving." Thinking & Reasoning, Vol. 10, No. 1, pp. 55-83.

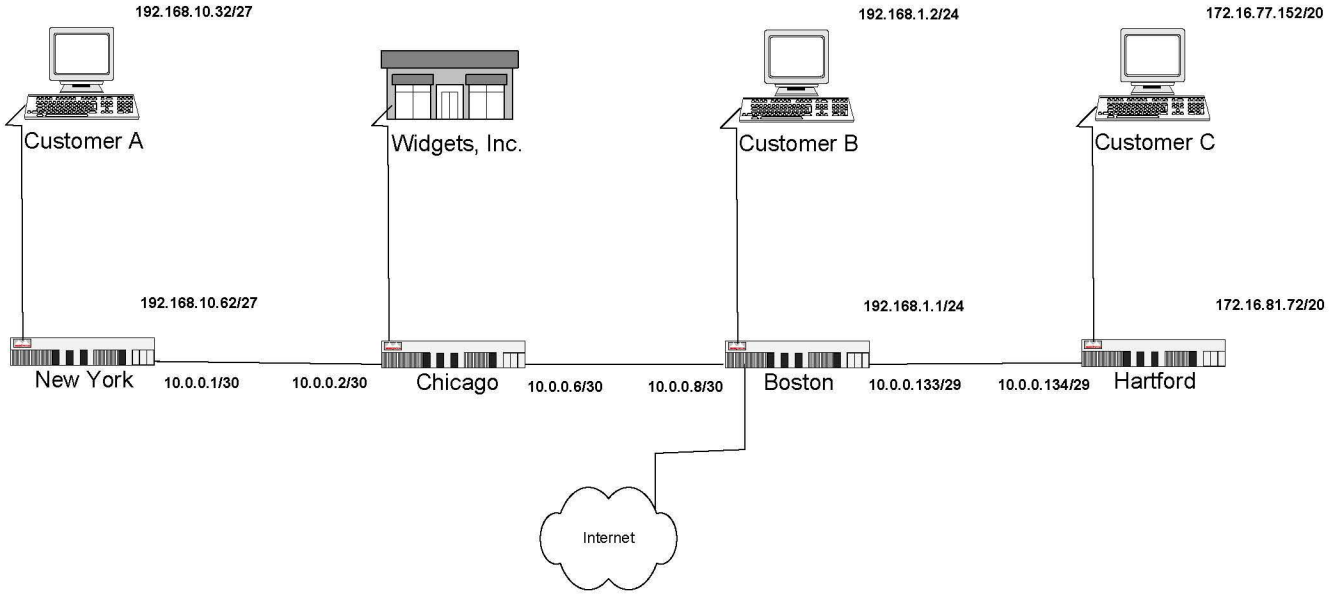
AUTHOR BIOGRAPHIES

Arnold Kamis is an Associate Professor in Information Systems and Operations Management at Suffolk University, Boston, Massachusetts. He received his Ph.D. in Information Systems from the Stern School of Business of New York University and his B.S. in Applied Mathematics (Computer Science) from Carnegie Mellon University. Arnold's research interests are in electronic commerce, decision support technologies, and human-computer interaction. His publications appear in *MIS Quarterly*, *International Journal of Cases on Electronic Commerce, The American Statistician, Information & Management, International Journal of Electronic Commerce, Communications of the ACM, Communications of the Association for Information Systems, The Database for Advances in Information Systems, Information Systems Frontiers* and *e-Service Journal*. Arnold serves as a chair for the HICSS Minitrack on Electronic Marketing and is the Web Site Editor for the Journal of Management Information Systems.

Beverly K. Kahn is an Associate Professor and chair of the Information Systems and Operations Management Department at the Sawyer Business School of Suffolk University in Boston, Massachusetts. She graduated from the University of Michigan with a Ph.D. and M.S. in Industrial and Operations Engineering and a B.A. in both Mathematics and Computer Science. Dr. Kahn's research concentrates on information quality, data warehouses, database design, and case studies. Her publications have appeared in leading journals such as *MIS Quarterly*, *Journal of Management Information Systems*, *Communications of the ACM*, and *The Database for Advances in Information Systems*. Beverly is affiliated with the MIT program on information quality.

Appendix A: sample Concrete and Abstract Problems

Concrete Problem (similar difficulty to Abstract Problem below)



Customer A has complained that they have not been able to access the Internet since their connection was installed. After some brief troubleshooting, support determined that they were unable to connect to the New York router. Why can't customer A connect to the New York router, and what would you recommend to fix the problem?

Abstract Problem (similar difficulty to Concrete Problem above)
Assume you are evaluating two different situations. In one, you have been assigned the network 172.16.0.0/16 and in the other, you have been assigned the network 10.0.0.0/20. In the first case, you are evaluating two IP addresses, 172.16.13.10/20 and 172.16.17.85/20, and in the second case another two, 10.0.0.89/30 and 10.0.0.90/30. You suspect that in one of these cases addresses are not within the same subnet. Is this true and if yes, why?

Appendix B: Keypad

Information Systems & Computing
Academic Professionals

STATEMENT OF PEER REVIEW INTEGRITY

All papers published in the Journal of Information Systems Education have undergone rigorous peer review. This includes an initial editor screening and double-blind refereeing by three or more expert referees.

ISSN 1055-3096