MIS Quarterly

RESEARCH ARTICLE

UNDERSTANDING MINDSHIFT LEARNING: THE TRANSITION TO OBJECT-ORIENTED DEVELOPMENT[1]

By: **Deborah J. Armstrong**
Department of Management Information Systems
College of Business
Florida State University
Tallahassee, FL 32306-1110
U.S.A.
djarmstrong@fsu.edu

Bill C. Hardgrave
Information Systems Department
Sam M. Walton College of Business
University of Arkansas
Fayetteville, AR 72701
U.S.A.
whardgra@uark.edu

Abstract

Information systems professionals increasingly face changes in their work environment. Some of these changes are incremental, but many require fundamental shifts in mindset (referred to as a mindshift). Within the domain of software development, previous research has determined that veteran developers experience difficulty making the transition to new forms of development. Although prior research has brought awareness to the problems caused by a mindshift and has provided some insight, it has not answered the question of why software developers have difficulty making the transition. This study begins to answer that question by positing and examining the mindshift learning theory (MLT). The MLT suggests that the degree of perceived novelty of the fundamental concepts that characterize the new mindset will impact learning. Specifically, concepts may be perceived as novel (i.e., not familiar to the learner), changed (i.e., similar to a known concept, but a different meaning in the new context), or carryover (i.e., known concept with a similar meaning in the new context). As an exemplar mindshift learning situation, this study explores the phenomenon in the context of software developers transitioning from traditional to object-oriented (OO) software development. Findings indicate that software developers had higher knowledge scores on the OO concepts they perceived as novel or carryover compared to those they perceived as changed. Thus, developers experienced detrimental interference from their existing traditional software development knowledge structure when trying to learn OO software development. The findings have implications for organizations and individuals as an understanding of mindshifts could mean an easier transition through decreased frustration and a more effective learning process.

Keywords: Software development, IS personnel, personnel training, learning theory, object-oriented

Introduction

One can think of a mindset as a distinct viewpoint that determines how an individual engages events or views reality (Culbert 1996). When essential or commonly held concepts fundamentally change, a revolutionary or quantum mindset change occurs. To describe these revolutionary changes in

[1]Lars Mathiassen was the accepting senior editor for this paper. Rob Fichman was the associate editor. Sandeep Purao and Juhani Iivari served as reviewers. The third reviewer chose to remain anonymous.

mindset, we have coined the term *mindshift*.[2] The information systems field commonly experiences such shifts. Examples include the shift from flat files to hierarchical to relational databases; from mainframe-centric computing to client-server to network-centric computing; and from traditional software development to object-oriented (OO) development. Each of these transitions requires not only a shift in tools and techniques, but also most importantly, a shift in the way IS professionals conceptualize and approach problems and solutions. Without this mindshift, many of the new mindset's advantages are lost because people employ the new techniques with old, incompatible ideas.

Even though the term mindshift is new, the phenomenon has long been of interest to researchers. Prior studies have established that learning during a mindshift *is* difficult (e.g., George 2002; Tapscott and Caston 1993), but have not directly addressed *why* it is so difficult. This study proffers a theory to answer why. Specifically, and as an instance of mindshift learning, this study explores the phenomenon in the context of software developers making the transition to OO development. We seek to understand the learning difficulties software developers encounter during the acquisition of fundamental OO concept (e.g., object) knowledge.

The proposed *mindshift learning theory* (MLT) provides insight into an individual's learning difficulties when initially shifting to a new mindset. The empirical portion of this study focuses on concept knowledge performance and is a necessary and important first step to ultimately identify why experienced developers find it difficult to learn a new software development approach. Our findings indicate that the MLT successfully explains aspects of mindshift learning in the given context (shift to OO development). Future research will incorporate additional portions of the theory to further explore the interplay of learning and knowledge structures.

The study of mindshift learning is significant because these mindshifts will impact not only individuals but also organizations. For the individual, customized training based on an understanding of mindshift learning could mean an easier transition through decreased frustration and a more effective learning process. For the organization, improving the training process may decrease organizational training costs, increase software quality, encourage wider adoption of the new technology, and, perhaps, ultimately increase employee satisfaction and retention. In a broader sense, the MLT may be applied to other mindshift learning situations, as the mindshifts that individuals face in the IS field will continue—most likely at an increasing rate (Bettis and Hitt 1995).

[2]Other names for the phenomenon include *punctuated change* (Gould and Eldredge 1977) and *second order change* (Bartunek and Moch 1987; Gash and Orlikowski 1991).

Context

The framework for classifying information systems development created by Iivari, Hirschheim, and Klein (1998, 2000-2001) establishes the milieu for this study. Divided into four hierarchical levels, the framework of paradigms, approaches, methodologies, and techniques provides a structure for understanding the IS development process (Iivari et al. 2000-2001). At the hierarchy's top, the *paradigm* is "the most fundamental set of assumptions adopted by a community that allows its members to share similar perceptions, and engage in commonly shared practices" (Hirschheim and Klein 1989, p. 1201).[3] Iivari et al. (2000-2001) assert that four paradigms exist in IS development: functionalism, social relativism, neohumanism and radical structuralism. (For a more detailed discussion of these paradigms, see Iivari et al. 2000-2001.) Within the framework, the second level in the hierarchy is the *approach*, which is a set of goals, guiding principles, and fundamental concepts that "drive interpretations and actions in IS development" (Iivari et al. 2000-2001, p. 186). The hierarchy's third level is the *methodology,* which is a set of goal-oriented procedures that guide the work and cooperation of the parties involved in building an IS application (Iivari et al. 2000-2001). At the lowest level are the *techniques*, which "consist of a well-defined sequence of basic operations, which permits the achievement of specific outcomes if executed correctly" (Iivari et al. 1998, p. 165).

When applying the IS development framework to the transition from traditional to OO development, the shift clearly occurs at the hierarchy's approach level. Iivari et al. (2000-2001) state that OO development and other specific forms of development, such as structured development, are distinct types of approaches that fall under the functionalist paradigm. Object-oriented software development is defined as "developing software that is centered on the concepts of cooperating objects and classes" (adapted from Booch 1994, pp. 36-37). Traditional software development is used in this context to represent any non-object-oriented software development approach (e.g., procedural, structured). Thus, if we contrast traditional and OO software development using this framework, it is appropriate to do so at the approach level.

[3]This definition is consistent with Burrell and Morgan's (1979) use of the term paradigm to describe the basic assumptions underlying coexistent theories of the social sciences, as opposed to Kuhn's (1970) use of the term to describe the historical development of the natural sciences, in which one theory or view of the world is supplanted by a different one.

When transitioning from one software development approach to another, the learning process may begin with individuals being introduced to the fundamental concepts that define the approach. This is consistent with the fundamental concepts feature of an approach as defined within the IS development framework. Our objective then is to understand the difficulty developers experience while learning the fundamental concepts (e.g., encapsulation, object, class) that underlay the new (OO software development) approach.

The Problem of Learning in Information Systems

An investigation of the extant IS literature from a learning perspective revealed three major research themes:[4] (1) successful software development education focuses on developing semantic knowledge first, and then syntactic; (2) experts create abstract mental representations of software development constructs, whereas novices create more concrete representations; and (3) software developers have difficulty making the transition from the traditional to the OO mindset.

The first theme deals with software development education. When teaching software development, the sequence of instruction impacts learning, within both the traditional and OO approaches. The common model for teaching software development incorporates a mixture of both semantics (language-independent general programming *concepts* and structures) and syntax (language-dependent details for carrying out actions) in the learning process (Mayer 1987; Shneiderman 1986; Shneiderman and Mayer 1979). General software development findings support the notion that students who learn semantics first and then syntax perform better than students who learn them simultaneously (Bayman and Mayer 1988; Dyck and Mayer 1989; Nowaczyk 1984; Spohrer and Soloway 1986). Within the OO mindset, it appears even more important to focus on learning the conceptual design and modeling aspects prior to introducing coding (Crews and Butterfield 2003; Hardgrave and Doke 2000; Nelson et al. 2002; Sheetz et al. 1997; Sircar et al. 2001). Thus, when communicating a new software development approach, initial training efforts should focus on the mindset's conceptual aspects.

The second theme addresses the cognitive representations or mental models of expert and novice software developers. Studies of the differences between novice and expert developers found that expert representations were more abstract and semantically focused (e.g., what a program does). Novice representations, on the other hand, were more concrete and syntactically focused (e.g., how a program functions) (Adelson 1981, 1984; Corritore and Wiedenbeck 1991; Guerin and Matthews 1990; McKeithen et al. 1981; Schenk et al. 1998; Shneiderman 1976; Soloway and Ehrlich 1984; Vitalari 1985; Wiedenbeck 1985, 1986).

The last theme deals directly with the transition from traditional to OO software development. Prior research indicates that software developers experienced in traditional development have difficulty making the transition to OO development compared to the performance of developers who have no traditional development experience (Nelson et al. 1997; Rosson and Alpert 1990; Rosson and Carroll 1990; Vessey and Conger 1994). Some researchers have found that expert software developers who are introduced to OO concepts will often fall back on their traditional knowledge (Detienne 1995; Gibson 1991; Manns and Nelson 1996; Pennington et al. 1995) and consequently acquire OO knowledge at a much slower rate.

Although prior IS research has brought awareness to the problem of mindshift learning and has provided some insight, it has not answered the question of why software developers have difficulty making the transition. This study addresses that gap by borrowing from schema theory and the concept of proactive interference.

Understanding the Learning Process

Knowledge Structures

A schema (originally proposed by Bartlett 1932) can be thought of as a framework of organized concepts which are the individual's representation of experience (Novak and Tyler 1977); "a cognitive structure that provides situational forecasts on which individuals rely" (Louis and Sutton 1991, p. 61); and a representation of a person's knowledge that includes both a set of domain-specific concepts and the relations among those concepts (Dorsey et al. 1999; Johnson-Laird 1983; Stasz et al. 1976). Other names for schema include knowledge structures, mental models, conceptual frameworks, cognitive structures, cognitive maps, and frames (Aarts and Dijksterhuis 2000; Bartlett 1932; Day et al. 2001; Dorsey et al. 1999; Gagne 1985; Glaser 1984, 1990; Johnson-Laird 1983; Kraiger et al. 1993; Rouse and Morris 1986). For consistency, we use the term *knowledge structure* hereafter to represent this concept.

[4]We do not claim to provide an exhaustive review of all of the literature in this domain; rather, a representation of key articles from each of the relevant research themes is provided.

Scholars in a variety of areas such as computer usage (Day et al. 2001), educational psychology (Sumfleth 1988), language acquisition (Schmidt 1988), expert systems (Mason and Tessmer 2000), medicine (Nash and Nash 2003), military decision making (Kraiger et al. 1995), negotiation (Bazerman et al. 2000), psychology (Hendrick 1983), and multiple areas in the marketing field (Aaker and Keller 1990; Boush and Loken 1991; Broniarczyk and Alba 1994; Lawson and Bhagat 2002; Luna and Peracchio 2002; Shimp et al. 1993) have studied the impact of knowledge structures on learning. The relationship between knowledge acquisition, knowledge structures, and concept knowledge consistently appears as a theme across these studies.

Concept Knowledge

As discussed earlier, individuals learn software development via the acquisition of semantic and syntactic knowledge, and semantic knowledge consists of concepts and the structure of those concepts (Mayer 1987; Shneiderman 1986; Schneiderman and Mayer 1979). Concepts have been defined as the actual ideas and information embodied in the knowledge (Ausubel 1963) or records of events or objects designated by a label (Novak 2002). As the individual is introduced to the concepts, he or she engages in cognitive processing[5] to learn the concepts.

In an incremental learning situation, as the individual is exposed to a concept, the knowledge structure perceived to most closely match the concept is activated. As the concept is learned, it is integrated into the activated knowledge structure. This modeling process is known as analogical learning (e.g., Gregan-Paxton and John 1997; Holyoak and Thagard 1995; Rumelhart and Norman 1981) or learning by metaphor (Carroll and Thomas 1982). As an individual's knowledge increases, he or she revises existing knowledge structures to organize that knowledge (Anderson 1982, 1995; Ausubel 1963, 1968; Kraiger et al. 1993; Piaget 1978, 1980; Rist 1989; Rumelhart and Norman 1981). For example, when a learner familiar with the Visual Basic.Net software development environment encounters an additional Visual Basic.Net concept (such as a ComboBox), incremental learning occurs. The individual activates his or her Visual Basic.Net software development knowledge structure and integrates the new concept into that existing structure.

In a mindshift learning situation, a learner will activate an existing knowledge structure that may only be partially appropriate for the new domain. The learner engages in cognitive processing and attempts to incorporate the concept into his or her activated knowledge structure. If the learner accesses knowledge (from the activated knowledge structure) that is inappropriate in the new domain, he or she experiences *proactive interference* because the existing body of knowledge interferes with the learning process (Underwood 1957). Essentially, proactive interference makes it difficult, if not impossible, to properly understand the concept within the context of the new mindset (Melton and Irwin 1940; Underwood 1957).

For example, when a learner familiar with COBOL encounters an unfamiliar Visual Basic.Net concept (such as a ComboBox), the individual activates his or her COBOL knowledge structure and attempts to integrate the new knowledge into that existing structure. As the new information is inconsistent with the COBOL knowledge structure, interference occurs. The new knowledge (Visual Basic.Net ComboBox concept) is incompatible with the existing knowledge structure (COBOL) and cannot be successfully incorporated. Subsequently, the learner creates a new Visual Basic.Net knowledge structure to incorporate the Visual Basic.Net ComboBox concept knowledge.

Cognitive Processing

As the examples illustrate, the cognitive processing required and the extent of proactive interference is not consistent across learning situations. Different learning situations will require more or less cognitive processing (i.e., incremental versus mindshift). Similarly, the extent of proactive interference is dependent on the situation (i.e., the knowledge being acquired and the knowledge structure activated).

In their work on cognitive processing, Louis and Sutton (1991) propose three categories of situations that engage the individual in conscious cognitive processing: novel, discrepant, and deliberate initiative. The *novel* category suggests a situation where an individual encounters an aspect not previously experienced, where something stands out of the ordinary, is unique, is unfamiliar, or previously unknown (Louis and Sutton 1991). They suggest that role transitions, such as promotions, transfers, and career changes, can be experienced as novel and trigger individuals to engage in active thinking. The *discrepant* category describes a situation where an unexpected failure, a disruption, or a significant difference exists between expectations and reality (Louis and Sutton 1991). For example, a performance review would fit into the discrepant category when a gap occurs between the individual's assessment of performance and the manager's.

[5]Cognitive processing occurs when an individual is involved in any type of information processing, thinking, or learning (Billett 1994).

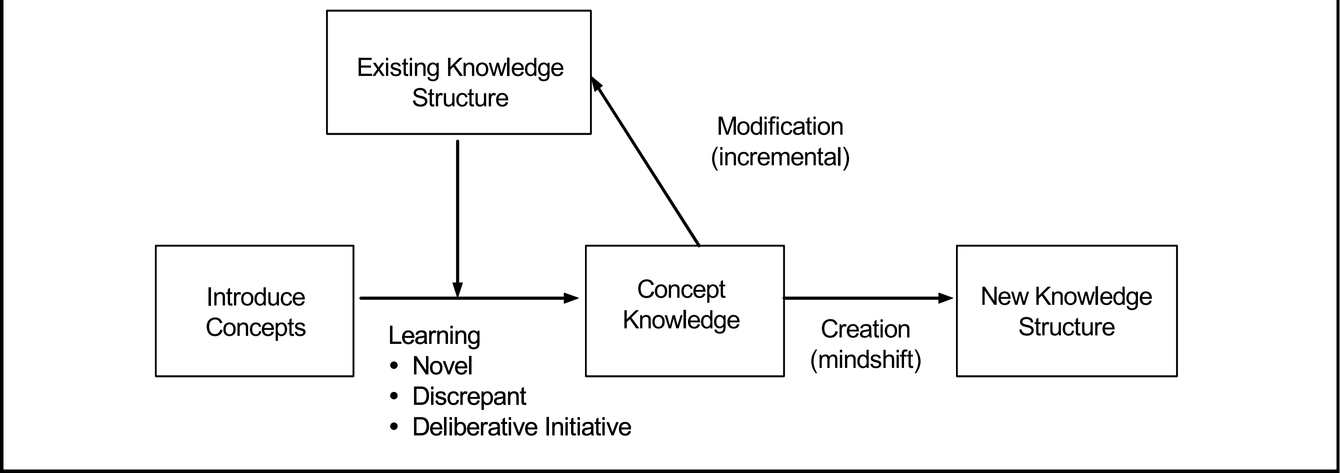

Figure 1. Base Theory of Mindshift Learning

The third category, *deliberate initiative*, describes an individual's response to a request for an increased level of attention, when asked to think, or while being explicitly questioned (Louis and Sutton 1991). They suggest that career planning triggers active thinking because individuals must stop and reflect on goals, resources, and opportunities.

Base Theory

From the preceding literature review, a base theory of mindshift learning is proposed (see Figure 1). Through the learning process, the introduction of concepts leads to concept knowledge, which, in turn, modifies an existing knowledge structure or creates a new one. However, the process will encounter positive, negative, or no transfer from an existing knowledge structure. The current study explores the effect of this transfer on the level of concept knowledge.

Refining the Theory

To contextualize the theory (the transition of traditional developers to the OO approach), it was first necessary to determine which concepts were fundamental to the OO approach. We utilized the work of Armstrong (2006) that identified, based on an extensive literature review, a set of nine concepts that are representative of the fundamental OO concepts. The OO concepts identified by Armstrong and used herein are abstraction, attribute, class, encapsulation, inheritance, method, message passing, object, and polymorphism.

From the work of Louis and Sutton (1991) and Armstrong (2006), it became apparent that a concept's origin has an important effect on understanding it. For example, some studies suggest that several of the OO approach's concepts are borrowed from the traditional software development approach (e.g., abstraction, attribute), whereas other concepts are new to OO development (e.g., class, inheritance). In addition, the literature is often contradictory in classifying the concepts. For example, some have said the encapsulation concept existed prior to the introduction of OO (Page-Jones and Weiss 1989) while others assert it first appeared in the Simula programming language (e.g., Booch 1994). Still others assert that encapsulation is just a new term for the information-hiding concept used in traditional development (e.g., Henderson-Sellers 1992; Yourdon et al. 1995). Thus, the concept origin (borrowed, new) could affect the cognitive processing requirements, and perhaps the level of proactive interference experienced by the learner.

While the categories proposed by Louis and Sutton (1991) seemed to roughly correspond to the different cognitive requirements of the OO concepts as suggested by the literature review (i.e., borrowed, new), it was necessary to ensure that the categories were appropriate for the domain under study. Subsequently, we employed an inductive approach to refine the theory (as suggested by Eisenhardt [1989] and Glaser and Strauss [1967]). Using induction, observations from the field are used to supplement or build theory (Eisenhardt 1989). In this case, OO experts were asked to assist with understanding the *learning* portion of Figure 1.

Three expert OO software developers independently performed a card sort of the nine OO concepts into the three categories suggested by Louis and Sutton: novel, discrepant, and deliberate initiative (the definition of each category was provided to the experts). However, the experts experienced difficulty sorting the concepts into the three categories. They clearly understood the novel category and had little difficulty placing concepts into that category (not identical concepts, but all three placed at least one concept into the novel category). They understood less clearly the discrepant and deliberate initiative categories. One developer worked his way through the concept cards and put everything he thought was novel into that category. He then tried to go through the remaining concept cards and place concepts into the discrepant category, and a third time into the deliberate initiative category. After all three rounds, several concept cards remained unsorted. Upon further discussion, the developer disclosed that the leftover concepts were not fitting into any of the three categories. Another developer placed all of the concept cards into one of the three categories, but when questioned as to why a certain concept was in the deliberate initiative category, for example, he responded, "I thought I had to put cards into each of the piles." The third developer also struggled with sorting the concepts into the given categories.

As a result of the failed sorting process, we asked each developer at the conclusion of the session to create his or her own categories for the concepts. While the specific concepts in the groupings differed, the experts created similar conceptual groupings of the concepts. Subsequently, three categories emerged which we call novel, changed, and carryover.

The *novel* category is defined as any concept not previously experienced or known (consistent with Louis and Sutton). The *changed* category is defined as any concept that had an existing meaning in traditional development, but has a new meaning in the OO development context. This category subsumes Louis and Sutton's discrepant and deliberate initiative categories. With the introduction of a changed concept, the learner will activate his or her existing traditional development knowledge structure. In this instance the introduced concept is cognitively close to an existing concept, but the concept usage is quite different. So, the learner will experience both the discrepant and deliberative initiative situations within the changed concepts. The learner will experience the discrepancy aspect because of the mismatch between the incoming information and the existing knowledge structure, and will experience the deliberate initiative aspect because he or she will have to think about the application of the concept within the new mindset. The *carryover* category contains those concepts that were originally defined in traditional development and continue to hold the same meaning in OO development (i.e., the borrowed concepts). Louis and Sutton did not conceptualize this category because their theory emphasized triggers for change, and the carryover category—being a case of non-change—was not included.

One final note on the concept categories: The taxonomy for each individual will be unique depending on his or her perception of the novelty of the OO concept (i.e., is it novel [high novelty], changed [some novelty], or carryover [low novelty]?). For example, the concept of encapsulation has been described in the literature as novel, carryover, and changed, demonstrating that the degree of perceived novelty may be high (novel), low (carryover), or somewhere in between (changed). Thus, how each individual perceives the novelty will determine how the information is cognitively processed.

From the base theory of mindshift learning (Figure 1) and the insights gained from the OO experts, we proffer a more refined theory incorporating the newly identified concept categories (see Figure 2) within the context of the transition from traditional to OO software development as a specific instance of the more general mindshift learning theory. In this case, the process of theory extension (i.e., taking a theory in one area and adapting it to another for application and modification) is used to create the theory (Osigweh 1989; Weick 1989). Specifically, we adapt the Louis and Sutton theory to the software development domain, expand the boundaries of their theory by the creation of the carryover category, and extend their theory via the modification of the discrepant and deliberative initiative categories into the changed category.

Theoretical Explanations and Hypotheses

From the literature, it seems clear that within the OO software development approach an emphasis on learning the conceptual aspects prior to the introduction of coding is key (e.g., Crews and Butterfield 2003; Sircar et al. 2001). As the individual encounters the fundamental OO concepts, he or she engages in cognitive processing to learn them. Some of the concepts introduced to the learner will be novel, and the introduction of these concepts adds information to his or her body of concept knowledge. The learner will not have any knowledge of these concepts in his or her existing traditional development knowledge structure from which an analogy can be drawn. The learner will integrate the novel concepts directly into his or her body of OO concept knowledge, thus expanding the individual's existing body of OO knowledge (or establishing a body of OO concept knowledge, if one did not already exist). Therefore, the existing traditional knowledge structure will have no influence on a learner's knowledge of the novel concept.

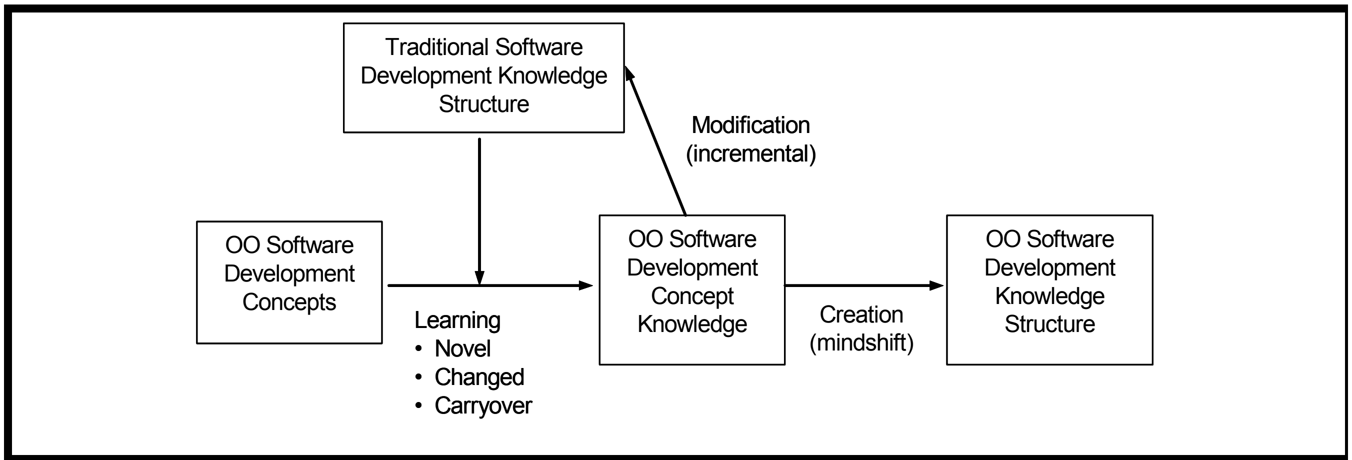
Traditional Software Development Knowledge Structure
Modification (incremental)
OO Software Development Concepts
Learning
• Novel
• Changed
• Carryover
OO Software Development Concept Knowledge
Creation (mindshift)
OO Software Development Knowledge Structure

Figure 2. Mindshift Learning Theory

Some of the concepts introduced to the learner will be changed. Changed concepts are defined as those OO concepts that existed in the traditional development approach but are used differently in the OO approach. A learner's existing knowledge structure is regarded as the most crucial factor influencing learning with the second being the ability to discriminate between existing information and new information (Ausubel 1963). For example, if the learner cannot distinguish between the existing and the new information, he or she may inappropriately apply the existing knowledge to the new domain. This lack of distinction between new ideas and previously learned information may account for some proactive interference (Anderson 1995). Therefore, the existing traditional knowledge structure will negatively influence a learner's knowledge of the changed concept.

Some concepts introduced to the learner are carried over into the new domain (Rumelhart and Norman 1981). A few studies found the positive transfer of learning to be contingent on the congruence of students' existing knowledge and the information that followed (Doran and Ngoi 1979; Wittrock and Cook 1975). When a concept is known and consistently used across the mindsets, it triggers no change in concept knowledge. For example, one could argue that the concept of abstraction—the act of representing reality in a simplified form by removing certain distinctions so that we can see the commonalities (Morris et al. 1999)—exists in both traditional and OO development. Assuming that concept is known and consistently used across both software development approaches, no change in concept knowledge takes place. Therefore, the traditional knowledge structure will positively influence a learner's knowledge of the carryover concept.

To summarize the theory, the introduced concepts can be perceived as *novel* (high novelty), *carryover* (low novelty), and *changed* (somewhere in between). The knowledge acquired for each OO concept will be influenced to a greater or lesser degree by the existing traditional development knowledge structure. Specifically, we expect changed concept knowledge to be influenced negatively by the existing traditional development knowledge structure such that carryover and novel concept knowledge will be higher due to the lack of proactive interference. Carryover concept knowledge, which will be positively influenced by the existing traditional development knowledge structure, should be higher than novel concept knowledge, which is not influenced by an existing knowledge structure. Therefore, the level of concept knowledge should be high when the degree of perceived novelty is near 0 percent (carryover concepts), decrease as the degree of novelty increases and proactive interference occurs (changed concepts), and then increase again as the degree of novelty approaches 100 percent (novel concepts).

From the previous discussion, we hypothesize that the degree of perceived novelty of an OO concept should affect learning, and ultimately, the developer's knowledge (as reflected by an objective score) of that OO concept such that

H1. A developer's OO concept knowledge score will have a U-shaped (curvilinear) relationship with the degree of perceived novelty.

Specifically, we expect differences in the concept knowledge acquired for the three concept categories to demonstrate the curvilinear relationship such that

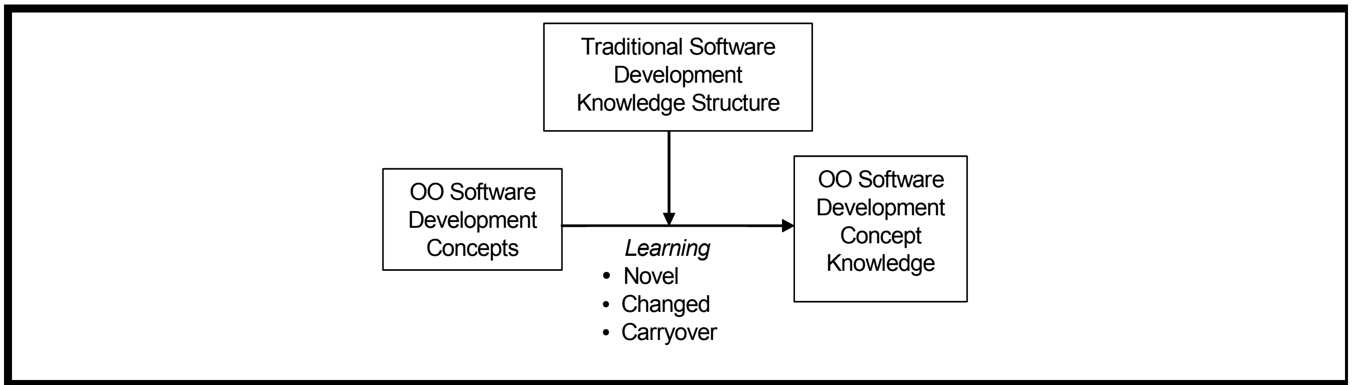
Traditional Software Development Knowledge Structure
OO Software Development Concepts
Learning
• Novel
• Changed
• Carryover
OO Software Development Concept Knowledge

Figure 3. Mindshift Learning Theory, Portion Under Study

> *H2. A developer's carryover concept knowledge score will be greater than his or her changed concept knowledge score.*
>
> *H3. A developer's carryover concept knowledge score will be greater than his or her novel concept knowledge score.*
>
> *H4. A developer's novel concept knowledge score will be greater than his or her changed concept knowledge score.*

As suggested by the hypotheses, the initial portion of the theory is examined in this study (see Figure 3). While this may not be the most vital part of the mindshift learning process, it is an important and necessary first step in developing an overall understanding of mindshift learning in this environment. If no difference exists in the OO concept knowledge of software developers across the concept categories, then there is no need to examine the full theory in its current form (Figure 2). Therefore, this study represents the first phase in examining the overall proposed theory.

Method

Subjects

A survey was distributed via mail or e-mail to a contact person within a set of organizations that agreed to participate in this research. The contact person, in turn, distributed the surveys to potential subjects. Survey eligibility required that each subject had experience with both traditional and OO software development. Unfortunately, but necessarily, this requirement severely limited the set of potential subjects. Overall, data were collected from 81 object-oriented (OO) software developers representing 16 companies from various industries (e.g., retailing, transportation, utilities, and government), with a response rate of 39 percent. The demographics of the sample can be found in Table 1.

Instrument Development

Because no instrument existed to measure OO conceptual knowledge or concept categorization, the necessary scales were developed as part of this study. The instrument's first section consisted of nine items and captured each concept's categorization by measuring its degree of perceived novelty on a scale from 0 to 100 percent. The second section consisted of 27 items that measured a subject's OO concept knowledge. The items were originally compiled from OO texts, reference books, and the authors' experience with the OO approach. Each concept was measured using three-item scales that were developed and validated as part of this research. The third section consisted of nine items that captured the level of perceived learning difficulty for each concept (on a seven-point Likert scale). See Appendix A for the complete instrument.

Instrument Validation and Pilot Study

Four peers evaluated the instrument, assessing face and content validity (see Straub 1989). As a result, we reworded items and, in some cases, dropped possibly ambiguous items, consistent with DeVellis's (2003) recommendations for scale development. We then conducted a focus group with 10 software developers (who possessed both traditional and OO software development experience) to evaluate the instrument and provide feedback. The developers gathered in a conference room at their organization to complete the instrument and

Table 1. Sample Demographics

Variable	Value
Gender	86% Male 14% Female
Age	
< 21	0%
21 – 30	44%
31 – 40	42%
41 – 50	9%
> 50	4%
Years of IS Experience	9.27
Years of Organizational Tenure	6.49
Years of Traditional Software Development Experience	6.78
Years of OO Software Development Experience	3.84

were debriefed after completing it. We subsequently deleted or reworded questions based on this additional feedback. Once completed, we distributed the instrument as detailed previously.

Results

To test the first hypothesis, we standardized the data and then analyzed it using polynomial regression in SPSS 12.0. The independent variable was the degree of perceived novelty, and the dependent variable was an individual's OO concept knowledge score. We scored each item on the OO concept knowledge scale as a one for a correct answer and a zero for an incorrect answer. With three questions per concept, each concept's minimum score was zero and its maximum score was three. We created a record for each concept for each participant for a total of 729 observations (81 respondents × 9 OO concepts). Each observation contained the respondent number, the concept identifier (e.g., object), the degree of perceived novelty, the concept knowledge score, and the level of perceived learning difficulty. To capture the curvilinear aspects of the model, the quadratic model is expressed as

$$\begin{aligned}&\textit{OO Concept Knowledge Score} = \\ &\alpha + \beta_1 \times \textit{Novelty} + \beta_2 \times \textit{Novelty}^2\end{aligned} \qquad (1)$$

The results indicate that the quadratic model provides an appropriate fit for the data (see Table 2 and Figure 4). From the results, we can see that both the degree of perceived novelty and the squared term are significant predictors of OO concept knowledge score. Taken together, the table and figure demonstrate that the curvilinear relationship more accurately represents the data, thus supporting H1.

Given the support for Hypothesis 1, the remaining hypotheses investigating the differences among concept categories can be examined. But, first, it was necessary to categorize the concepts (by subject) so that a comparison by category could be conducted (to test H2 through H4). To determine concept category (based on degree of perceived novelty), a visual inspection of the degree of perceived novelty histograms at the concept level revealed consistent patterns of responses (per concept), suggesting category cutoffs of 25 percent and 75 percent. Specifically, concepts identified in the 0 to 24 percent novel range were categorized as carryover; those in the range of 25 to 75 percent were categorized as changed; and those in the range 76 to 100 percent were categorized as novel. For example, if a respondent found an object concept 80 percent novel, then it was categorized as novel, but if she found it only 10 percent novel, then it was categorized as carryover. Because the use of histograms could be considered arbitrary, we then conducted a sensitivity analysis to determine the impact of various combinations of cutoff points. We found that while the actual mean OO concept knowledge score of the categories differed slightly (but not significantly), any cutoff points ranging from 20 to 30 percent for the lower bound and 65 to 80 percent for the upper bound produced the same pattern of results. Ultimately, we used the original 25/75 split. Thus, while any cutoff points could be argued as arbitrary, we believe that the cutoffs used in this analysis are robust and reflective of the nature of the data. The categories are illustrated in Figure 4.

Table 2. Regression Results for Degree of Perceived Novelty

A. Quadratic Model				
	Analysis of Variance			
	DF	**Sum of Squares**		**Mean Square**
Regression	2	56.117		28.059
Residuals	724	671.869		0.928
	F = 30.238		Sig. F = 0.000	
	Variables in the Equation			
Variable	**Beta**	**Std. Error.**	**T**	**Sig. T**
(Constant)	.000	.036	.006	.995
DegNov	–1.029	.150	–6.838	.000
DegNov2	1.131	.150	7.522	.000
B. Linear Model				
	Analysis of Variance			
	DF	**Sum of Squares**		**Mean Square**
Regression	1	3.608		3.608
Residuals	725	724.379		.999
	F = 3.611		Sig. F = 0.058	
	Variables in the Equation			
Variable	**Beta**	**Std. Error.**	**T**	**Sig. T**
(Constant)	.000	.037	.006	.995
DegNov	.070	.038	1.900	.058

Dependent variable: OO concept knowledge score
Independent variables: DegNov (degree of perceived novelty), DegNov2 (degree of perceived novelty squared)

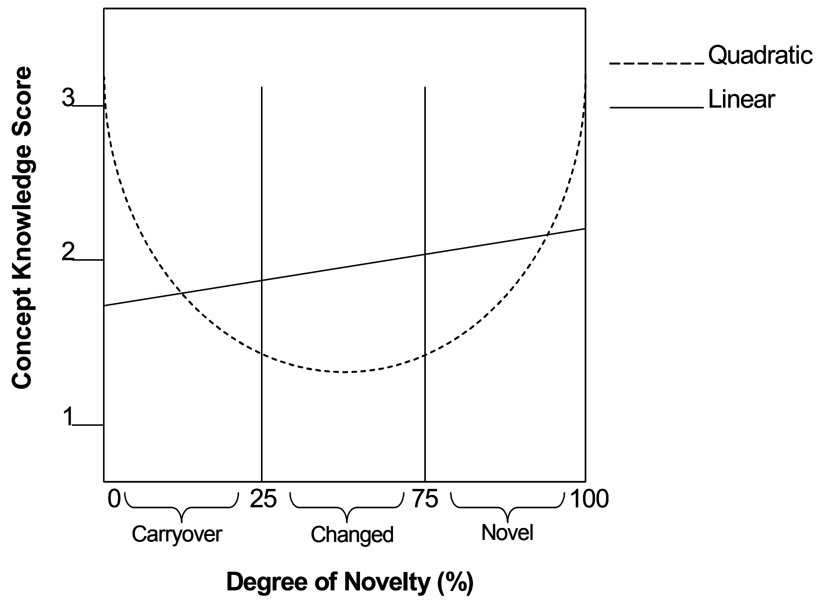

Figure 4. Regression Curve

Table 3. Object Concept Categorization

	Novel	Changed	Carryover
Subject 1	2		
Subject 2		1	
Subject 3			3
Subject 4			2
Subject 5		0	
Mean Object Concept Knowledge Score	*2.0*	*0.5*	*2.5*

Table 4. OO Concept Knowledge Scores by Category

Concept	Novel	Changed	Carryover
Abstraction	2.08 (n = 39)	1.70 (n = 23)	2.32 (n = 19)
Attribute	1.78 (n = 9)	1.47 (n = 34)	2.55 (n = 38)
Class	2.62 (n = 47)	2.16 (n = 25)	2.33 (n = 9)
Encapsulation	2.75 (n = 36)	1.27 (n = 33)	2.83 (n = 12)
Inheritance	2.22 (n = 67)	1.55 (n = 11)	2.00 (n = 3)
Message Passing	2.17 (n = 12)	1.59 (n = 32)	2.08 (n = 37)
Method	1.50 (n = 8)	1.37 (n = 27)	1.98 (n = 46)
Object	2.36 (n = 47)	2.04 (n = 27)	2.29 (n = 7)
Polymorphism	2.14 (n = 64)	1.50 (n = 12)	1.80 (n = 5)
Overall Mean	***2.29 (n = 329)***	***1.62 (n = 224)***	***2.24 (n = 176)***

Cell Values: mean (sample size)

Note: The sample size in each cell indicates the number of times the concept was placed in that category. For example, for the *Abstraction* concept, 39 individuals categorized it as novel, 23 as changed, and 19 as carryover. For the overall mean, the sample size indicates the total number of times something was placed in that category. Each row's sample size is 81 (for the number of respondents); the total table sample size is 729 (81 respondents × 9 concepts).

Table 5. Concept Categorization of Test Statistics

	Analysis of Variance		
	DF	**Sum of Squares**	**Mean Square**
Between Groups	2	66.772	33.386
Within Groups	726	543.228	.748
	F = 44.619	Sig. F = 0.000	

		t Tests		
Hypothesis	**Concept Comparison**	**T**	**df**	**Sig.**
2	Carryover > Changed	–7.007	388.289	.000
3	Carryover > Novel	.603	350.366	.547
4	Novel > Changed	8.730	443.897	.000

Once we determined the cutoff points, we placed the OO concept knowledge scores into the appropriate categories. For example, if a person perceived "object" as a novel concept then his or her score on the object concept (0 to 3) would be added to the novel object concept knowledge score. If another person perceived object as a carryover concept, then his or her concept knowledge score would be added to the carryover object concept knowledge score. See Table 3 for the object concept example.

This procedure was followed for each subject for each concept and then aggregated yielding means for each category (as shown in Table 4). The mean for the novel concepts was 2.29, the changed concepts 1.62, and the carryover concepts 2.24.

A one-way analysis of variance with contrasts was used to determine if the OO concept knowledge scores for the three categories (novel, changed, and carryover) were statistically significantly different. As shown in Table 5, two of the three hypotheses were supported. The carryover concept mean (2.24) was significantly greater than the changed concept mean (1.62), thus supporting H2. The carryover concept mean (2.24) was not significantly different (greater) than the novel concept mean (2.29). Thus, H3 was not supported. Lastly, the novel concept mean (2.29) was significantly greater than the changed concept mean (1.62), thus supporting H4.

Discussion

The proposed MLT provides a plausible explanation for the learning difficulties encountered during a mindshift and has implications for both theory and practice. The theory suggests that an individual's perception of each OO concept's novelty impacts the learning process. Specifically, and as hypothesized, the changed concepts prove most problematic for the learner such that developers performed better on concepts perceived to be more (novel) or less (carryover) novel (compared to changed). We argue that existing traditional software development knowledge interfered with the learning of some OO development concepts. This occurred because the concepts perceived as changed are cognitively close to traditional development concepts that the learner has used (possibly for a long time). Conversely, when the learner perceives she is learning novel OO concepts, she has no existing traditional development knowledge from which to draw or use as an analogy. When learning the carryover OO concepts, the learner is very familiar with the concepts and their usage from the traditional development approach.

Contrary to hypothesis 3, we did not find a significant difference in OO concept knowledge score between novel and carryover concepts. One possible explanation for the lack of difference between these two categories is, over time, a respondent's novel concept knowledge improved to the point that it was equal to his or her knowledge of the carryover concepts. Specifically, the years of OO experience may have been a factor. To explore this variable further, we first examined the possibility that the developers' years of OO experience affected the OO concept knowledge scores. A correlation analysis between years of OO experience and concept knowledge score did, indeed, reveal a small, but significant, result (r = .130; p = .001). Due to the significant correlation representing a rival explanation to our overall findings (i.e., that OO concept knowledge score varies by degree of perceived novelty), we added years of OO experience to the existing regression equation (see equation 1) as a control variable. As shown in Table 6, years of OO experience has a significant relationship with concept knowledge score, but does not attenuate the importance of the degree of perceived novelty in the regression. We further tested for an interaction effect between years of OO experience and degree of perceived novelty and found it nonsignificant (Table 6). Thus, while years of OO experience has a small positive impact on OO concept knowledge score, the degree of perceived novelty provides the major influence on OO concept learning.

To further investigate years of OO experience relative to OO concept knowledge score, we examined the correlations by concept category (novel, changed, carryover). In other words, do OO concept knowledge scores improve over time (with years of experience) within each of the concept categories? As shown in Table 7, correlations between OO concept knowledge score and years of OO experience within the novel and carryover categories are borderline significant (p = .102 and .059, respectively), while the changed category is not significant (p = .232). Thus, it appears that the OO concept knowledge for the novel and carryover categories may improve slightly over time, whereas the OO concept knowledge for the changed category does not. This finding has important implications as it would suggest that the problems caused by proactive interference tend to persist over time.

Next, we turned our attention to the possible interplay between years of OO experience and degree of perceived novelty at the concept level. One might expect that a developer with many years of OO experience may view OO concepts as less novel whereas a developer with less OO experience may view OO concepts as more novel. To rule out the possibility that a respondent's years of OO experience influenced his or her assessment of the degree of perceived novelty of the OO concepts, we examined the correlation between the two by concept (i.e., object, polymorphism, class, etc.). As shown in Table 8, none of the correlations are significant. Thus, one's assessment of the degree of perceived novelty of the OO concepts does not appear to vary by years of OO experience.

Table 6. Regression Results for Years of OO Experience

	Analysis of Variance			
	DF	Sum of Squares	Mean Square	
Regression	4	72.456	18.114	
Residuals	686	614.604	.896	
	F = 20.218		Sig. F = 0.000	
	Variables in the Equation			
Variable	**Beta**	**Std. Error.**	**T**	**Sig. T**
(Constant)	.033	.036	.923	.357
DegNov	–.980	.161	–6.071	.000
DegNov2	1.162	.151	7.691	.000
YrsOO	.161	.064	2.514	.012
YrsOODegNov	–.081	.086	–.935	.350

Dependent variable: OO concept knowledge score
Independent variables: DegNov (degree of perceived novelty), DegNov2 (degree of perceived novelty squared), YrsOO (years of OO experience), YrsOODegNov (years of OO experience × degree of perceived novelty)

Table 7. Correlation Analysis: OO Concept Knowledge Score and Years of OO Experience by Concept Category

		OO Concept Knowledge Score	
		Pearson Correlation	Sig. (2-tailed)
Years of OO Experience	Novel (n = 301)	.094	.102
	Changed (n = 220)	.081	.232
	Carryover (n = 172)	.144	.059

Table 8. Correlation Analysis: Years of OO Experience and Degree of Perceived Novelty by Concept

Concept	Pearson Correlation	Sig.
Abstraction	.132	.254
Attribute	–.152	.188
Class	.087	.452
Encapsulation	–.079	.496
Inheritance	.082	.476
Message Passing	.030	.800
Method	–.051	.657
Object	–.009	.935
Polymorphism	–.088	.391

*Correlation between years of OO experience and degree of perceived novelty.

Table 9. Regression Results with Level of Perceived Learning Difficulty

	Analysis of Variance		
	DF	Sum of Squares	Mean Square
Regression	4	57.665	14.416
Residuals	717	661.384	.922
	F = 15.629	Sig. F = 0.000	

	Variables in the Equation			
Variable	Beta	Std. Error.	T	Sig. T
(Constant)	.005	.036	.130	.896
DegNov	–.999	.153	–6.512	.000
DegNov2	1.130	.159	7.111	.000
Diff	–0.28	.084	–.339	.735
YrsDegNovDiff	–.021	.119	–.174	.862

Dependent variable: OO concept knowledge score
Independent variables: DegNov (degree of perceived novelty), DegNov2 (degree of perceived novelty squared), Diff (level of perceived learning difficulty), DegNovDiff (degree of perceived novelty × level of perceived learning difficulty)

Another potential influence on OO concept knowledge score may be a particular concept's difficulty or complexity. For example, one could hypothetically say that polymorphism is a more difficult (complex) concept than attribute. Therefore, we would expect a developer to score higher on the OO concept knowledge scale for the attribute concept than for the polymorphism concept, regardless of the degree of perceived novelty. Unfortunately, no objective measure of OO concept complexity exists. Therefore, to examine the possibility that the concepts' difficulty influenced the differences in OO concept knowledge scores, we asked respondents to recall when they first learned each concept and indicate the level of perceived learning difficulty (as a subjective measure of complexity). Overall, no significant correlation existed between the level of perceived learning difficulty and OO concept knowledge score ($r = –.035$, $p = .346$). Next, the difficulty measure was added to the regression to test for an interaction effect (with degree of perceived novelty). As shown in Table 9, there is no relationship between the level of perceived learning difficulty and the OO concept knowledge score. While we find this result encouraging, future research should consider the development of objective measures of OO concept complexity and the incorporation thereof as a control variable in this theory.

Limitations and Future Research

In this study, we used a cross-sectional, post-learning design. With this design, it was not possible to test directly the impact of experience over time on OO concept knowledge. Ideally, we would like to test an individual's knowledge at various points in time (i.e., using a within-subjects design rather than a between-subjects design). The perceived novelty measure, while meant to be current, may be affected by the passage of time (e.g., a developer's perception of novelty 5 years after learning a concept may be different than when it was first introduced). Although we found no correlation between the degree of perceived novelty and years of OO experience, the only way to truly rule out the effect of time would be to conduct a longitudinal study. Using both current (OO concept knowledge scores) and retrospective (level of perceived learning difficulty) measures may have also confounded the results by intermingling two time periods. Lastly, because we measured both concept knowledge and perceived novelty cross-sectionally, one could speculate that the level of concept knowledge may influence the degree of perceived novelty (as opposed to the degree of perceived novelty influencing the OO concept knowledge score). Overall, a longitudinal, within-subjects design which follows a learner throughout the process would attenuate the aforementioned shortcomings.

Future research focused on broader processes might consider other scenarios within OO software development. For example, research could address the situation in which an individual is introduced to concept knowledge from a new mindset (e.g., OO software development) but has no existing related knowledge structure (e.g., no software development experience) upon which to draw. At the other extreme, research could address the case in which a learner introduced to concept knowledge from a new mindset (e.g., OO software development) draws on multiple existing software development knowledge structures (e.g., structured, data-oriented). In general, questions such as from which (if any) existing knowledge structure domain(s) the learner would draw, and how many different existing knowledge structures would be utilized, could be of interest.

In addition to providing new explanations as to why mindshift learning is so difficult within the transition from the traditional to the OO approach, the theory has the potential to generalize to other mindshift learning situations. Whether the shift under study is within the software development field, information systems, or other aspects of organizations, the principles identified in the MLT may improve understanding of the difficulties individuals will encounter during these transitions.

Implications

This research has significant implications for managers because the advances in tools, techniques, and methods will continue, and most likely increase in their frequency of introduction (Bettis and Hitt 1995). Thus, organizations will likely require software developers to make frequent mindshifts. Those organizations already committed to the shift from traditional to OO development may use the MLT as a basis for decision-making regarding the emphasis of retraining resources. Those considering the switch to OO development may use it to ease management concerns or increase commitment.

Organizations can adapt training programs to place differential emphasis on the OO concepts for different developers during the training process. For example, computer-based training modules with different perspectives could be developed for each concept (such as abstraction-novel, abstraction-changed, abstraction-carryover). After introducing the key concepts to students, an on-line diagnostic test (perhaps similar to the instrument developed here) could determine potential problem areas (i.e., perceived changed concepts). The organization could then provide customized training modules based on the results of the test.

In a face-to-face format, an understanding of the audience could help an instructor tune the training (post-introductory sessions). Even if training is not customized to each individual, but instead to a large portion of the class, that understanding may help overall learning effectiveness. For example, if a portion of the class perceived a concept as changed, an instructor could modify her message to include a participatory discussion of the concept's usage in the previous domain, or perhaps a discussion to evoke the concept's differences between the previous domain usage and its current domain usage (see Nelson et al. 2002). This may allow the learner to more accurately incorporate the concept into his or her knowledge structure of the new domain. Understanding and utilizing the concept categorization may attenuate the proactive interference and increase training effectiveness.

Conclusion

The motivation for this research was to understand the difficulties individuals experience when they are involved in a mindshift learning situation. Within the software development domain, previous research has determined that mindshift learning is more difficult than incremental learning, although it has not discovered why. To answer that question, this study posits and examines the mindshift learning theory within the context of the transition from traditional to OO software development. Specifically, we describe and test how the degree of perceived novelty (novel, changed, and carryover) of the fundamental OO concepts influences OO concept knowledge. Our findings indicate that individuals had higher scores on the OO concepts they perceived as novel or carryover than those they perceived as changed.

An understanding of the learning processes involved in transitioning from one mindset to another could aid change management initiatives, instructional design, and the learning process, and ultimately increase the benefits of the new mindset. From a theoretical perspective, questions of cognitive processing, knowledge structures, and proactive interference are important to our understanding of mindshift learning and should continue to be explored.

Acknowledgments

The authors would like to thank the senior editor, associate editor, and three anonymous reviewers for their helpful and insightful comments and suggestions for this paper. The authors would also like to acknowledge the Information Technology Research Institute at the Sam M. Walton College of Business at the University of Arkansas for assistance in the funding of this project. Finally, the first author would like to acknowledge her hero, Ken Armstrong.

References

Aaker, D., and Keller, K. L. "Consumer Evaluations of Brand Extensions," *Journal of Marketing* (54:1), January 1990, pp. 27-41.

Aarts, H., and Dijksterhuis, A. "Habits as Knowledge Structures: Automaticity in Goal-Directed Behavior," *Journal of Personality and Social Psychology* (78:1), January 2000, pp. 53-63.

Adelson, B. "Problem Solving and the Development of Abstract Categories in Programming Languages," *Memory and Cognition* (9:4), 1981, pp. 422-433.

Adelson, B. "When Novices Surpass Experts: The Difficulty of a Task May Increase with Expertise," *Journal of Experimental Psychology: Learning, Memory and Cognition* (10:3), 1984, pp. 483-495.

Anderson, J. R. "Acquisition of Cognitive Skill," *Psychological Review* (89), 1982, pp. 369-406.

Anderson, J. R. *Learning and Memory*, Wiley, New York, 1995.

Armstrong, D. J. "The Quarks of Object-Oriented Development," *Communications of the ACM* (49:2), 2006, pp. 123-128.

Ausubel, D. P. *Educational Psychology: A Cognitive View*, Holt, Rinehart and Winston, Inc, New York, 1968.

Ausubel, D. P. *The Psychology of Meaningful Verbal Learning*, Grune and Stratton, New York, 1963.

Bartlett, F. C. *Remembering: A Study in Experimental and Social Psychology*, University Press, Cambridge, England, 1932.

Bartunek, J. M., and Moch, M. K. "First-Order, Second-Order, and Third-Order Change and Organization Development Interventions: A Cognitive Approach," *The Journal of Applied Behavioral Science* (23:4), December 1987, pp. 483-501.

Bayman, P., and Mayer, R. E. "Using Conceptual Models to Teach BASIC Computer Programming," *Journal of Educational Psychology* (80:3), 1988, pp. 291-298.

Bazerman, M. H., Curhan, J. R., Moore, D. A., and Valley, K. L. "Negotiation," *Annual Review of Psychology* (51), 2000, pp. 279-315.

Bettis, R. A., and Hitt, M. A. "The New Competitive Landscape," *Strategic Management Journal* (16), Summer 1995, pp. 7-19.

Billett, S. "Situating Learning in the Workplace: Having Another Look at Apprenticeships," *Industrial and Commercial Training* (26:11), 1994, pp. 9-16.

Booch, G. *Object Oriented Analysis and Design with Applications*, Benjamin/Cummings Publishing Company, Inc., Redwood City, CA, 1994.

Boush, D., and Loken, B. "A Process Tracing Study of Brand Extension Evaluation," *Journal of Marketing Research* (28:1), February 1991, pp.16-28.

Broniarczyk, S., and Alba, J. "The Importance of the Brand in Brand Extension," *Journal of Marketing Research* (31:2), May 1994, pp. 214-228.

Burrell, G., and Morgan, G. *Sociological Paradigms and Organizational Analysis*, Heineman, London, 1979.

Carroll, J. M., and Thomas, J. C. "Metaphor and the Cognitive Representation of Computing Systems," *IEEE Transactions on Systems, Man and Cybernetics*, SMC (12:2), 1982, pp. 107-116.

Corritore, C. L., and Wiedenbeck, S. "What Do Novices Learn During Program Comprehension?," *International Journal of Human Computer Interaction* (3:2), 1991, pp. 199-123.

Crews, T., and Butterfield, J. "Improving the Learning Environment in Beginning Programming Classes: An Experiment in Gender Equity," *Journal of Information Systems Education* (14:1) 2003, pp. 69-76.

Culbert, S. *Mind-Set Management*, Oxford University Press, New York, 1996, pp. 14-21.

Day, E. A., Arthur Jr., W., and Gettman, D. "Knowledge Structures and the Acquisition of a Complex Skill," *Journal of Applied Psychology* (86:5), October 2001, pp. 1022-1033.

Detienne, F. "Design Strategies and Knowledge in Object-Oriented Programming: Effects of Experience," *Human Computer Interaction* (10:2/3), 1995, pp. 129-169.

DeVellis, R. F. *Scale Development: Theory and Applications* (2nd ed.), Sage Publications Inc., London, 2003.

Doran, R., and Ngoi, M. K. "Retention and Transfer of Selected Science Concepts in Elementary School Students," *Journal of Research in Science Teaching* (16:3), 1979, pp. 211-216.

Dorsey, D. W., Campbell, G. E., Foster, L. L., and Miles, D. E. "Assessing Knowledge Structures: Relations with Experience and Post-Training Performance," *Human Performance* (12), 1999, pp. 31-57.

Dyck, J. L., and Mayer, R. E. "Teaching for Transfer of Computer Program Comprehension Skill," *Journal of Educational Psychology* (81:1), March 1989, pp. 16-24.

Eisenhardt, K. M. "Building Theories from Case Study Research," *Academy of Management Review* (14:4), October 1989, pp. 532-551.

Gagne, R. M. *The Conditions of Learning, and Theory of Instruction* (4th ed.), Holt, Rinehart and Winston, Inc., Fort Worth, TX, 1985.

Gash, D. C., and Orlikowski, W. J. "Changing Frames: Toward an Understanding of Information Technology and Organizational Change," *Academy of Management Proceedings*, 1991, pp. 189-193.

George, F. "Transitioning to a Two-Crew Cockpit: Two Heads are (Almost) Always Better Than One, But Those New to Shared Responsibilities Have to Work for the Benefits," *Business and Commercial Aviation* (91:1), July 2002, pp. 64-68..

Gibson, E. "Flattening the Learning Curve: Educating Object-Oriented Developers," *Journal of Object Oriented Programming* (3:6), February 1991, pp. 24-29.

Glaser, R. "Education and Thinking: The Role of Knowledge," *American Psychologist* (39:2), February 1984, pp. 93-104.

Glaser, R. "The Reemergence of Learning Theory within Instructional Research," *American Psychologist* (45:1), January 1990, pp. 29-40.

Glaser, R., and Strauss, A. *The Discovery of Grounded Theory*, Aldine Press, Chicago, IL, 1967.

Gould S. J., and Eldredge, N. "Punctuated Equilibria: The Tempo and Mode of Evolution Reconsidered," *Paleobiology* (3:2), 1977, pp. 115-151.

Gregan-Paxton, J., and John, D. R. "Consumer Learning by Analogy: A Model of Internal Knowledge Transfer," *Journal of Consumer Research* (24:3), December 1997, pp. 226-245.

Guerin, B., and Matthews, A. "The Effects of Semantic Complexity on Expert and Novice Computer Program Recall and Comprehension," *The Journal of General Psychology* (117:4), October 1990, pp. 379-389.

Hardgrave, B. C., and Doke, E. R. "Cobol in an Object-Oriented World: A Learning Perspective," *IEEE Software* (17:2), March 2000, pp. 26-29.

Henderson-Sellers, B. *A Book of Object-Oriented Knowledge*, Prentice Hall, Englewood Cliffs, NJ, 1992.

Hendrick, H. W. "Pilot Performance Under Reversed Control Stick Conditions," *Journal of Occupational and Organizational Psychology* (56:4), 1983, pp. 297-301.

Hirschheim, R., and Klein, H. K. "Four Paradigms of Information Systems Development," *Communications of the ACM* (32:10), October 1989, pp. 1199-1216.

Holyoak, K. J., and Thagard, P. R. *Mental Leaps: Analogy in Creative Thought*, MIT Press, Cambridge, MA, 1995.

Iivari, J., Hirschheim, R., and Klein, H. K. "A Dynamic Framework for Classifying Information Systems Development Methodologies and Approaches," *Journal of Management Information Systems* (17:3), Winter 2000-2001, pp. 179-218.

Iivari, J., Hirschheim, R., and Klein, H. K. "A Paradigmatic Analysis Contrasting Information Systems Development Approaches and Methodologies," *Information Systems Research* (9:2), June 1998, pp. 164-193.

Johnson-Laird, P. N. *Mental Models*, Harvard University Press, Cambridge, MA, 1983.

Kraiger, K., Ford, K. J., and Salas, E. "Application of Cognitive, Skill Based and Affective Theories of Learning Outcomes to New Methods of Training Evaluation," *Journal of Applied Psychology* (78:2), April 1993, pp. 311-328.

Kraiger, K., Salas, E., and Cannon-Bowers, J. A. "Measuring Knowledge Organization as a Method for Assessing Learning During Training," *Human Factors* (37:4), 1995, pp. 804-816.

Kuhn, T. S. *The Structure of Scientific Revolutions* (2nd ed.), University of Chicago Press, Chicago, IL, 1970.

Lawson, R., and Bhagat, P. S. "The Role of Price Knowledge in Consumer Product Knowledge Structures," *Psychology and Marketing* (19:6), June 2002, pp. 551-560.

Louis, M. R., and Sutton, R. I. "Switching Cognitive Gears: From Habits of Mind to Active Thinking," *Human Relations* (44:1), January 1991, pp. 55-76.

Luna, D., and Peracchio, L. A. "Uncovering the Cognitive Duality of Bilinguals Through Word Association," *Psychology and Marketing* (19:6), June 2002, pp. 457-476.

Manns, M. L., and Nelson, H. J. "Retraining Procedure-Oriented Developers: An Issue of Skill Transfer," *Journal of Object-Oriented Programming*, November-December, 1996, pp. 6-10.

Mason, S. D., and Tessmer, M. A. "Expert Systems as a Mindtool to Facilitate Mental Model Learning," *Educational Technology, Research and Development* (48:4), 2000, pp. 43-63.

Mayer, R. E. "Cognitive Aspects of Learning and Using a Programming Language," in *Interfacing Thought: Cognitive Aspects of Human-Computer Interaction*, J. M. Carroll (ed.), MIT Press, Cambridge, MA, 1987, pp. 61-79.

McKeithen, K., Reitman, J., Rueter, H., and Hirtle, S. "Knowledge Organization and Skill Differences in Computer Programmers," *Cognitive Psychology* (13), 1981, pp. 307-325.

Melton, A. W., and Irwin, J. M. "The Influence of the Degree of Interpolated Learning on Retroactive Inhibition and the Overt Transfer of Specific Responses," *American Journal of Psychology* (53), 1940, pp. 173-203.

Morris, M. G., Speier, C., and Hoffer, J. A. "An Examination of Procedural and Object-Oriented Systems Analysis Methods: Does Prior Experience Help or Hinder Performance?" *Decision Sciences* (30:1), Winter 1999, pp. 107-137.

Nash, J. D., and Nash, J. M. "A Structural Representation of Migraine Diagnostic Criteria: The Experts View," *Headache* (43:4), 2003, pp. 322-329.

Nelson, H. J., Armstrong, D. J., and Ghods, M. "Teaching Old Dogs New Tricks," *Communications of the ACM* (45:10), 2002, pp. 132-137.

Nelson, H. J., Irwin, G., and Monarchi, D. E. "Journeys Up the Mountain: Different Paths to Learning Object-Oriented Programming," *Accounting, Management and Information Technology* (7:1), January 1997, pp. 53-85.

Novak, J. D. "Meaningful Learning: The Essential Factor for Conceptual Change in Limited or Inappropriate Propositional Hierarchies Leading to Empowerment of Learners," *Science Education* (86:4), July 2002, pp. 548-571.

Novak, J. D., and Tyler, R. W. *A Theory of Education,* Cornell University Press, Ithaca, NY, 1977.

Nowaczyk, R. H. "The Relationship of Problem Solving Ability and Course Performance Among Novice Programmers," *International Journal of Man-Machine Studies* (21), 1984, pp. 149-160.

Osigweh, C. A. B. "Concept Fallibility in Organizational Science," *Academy of Management Review* (14:4), 1989, pp. 579-594.

Page-Jones, M., and Weiss, S. "Synthesis: An Object-Oriented Analysis and Design Method," *American Programmer* (2:7/8), 1989, pp. 64-67.

Pennington, N., Lee, A., and Rehder, B. "Cognitive Activities and Levels of Abstraction in Procedural and Object-Oriented Design," *Human Computer Interaction* (10), 1995, pp. 171-226.

Piaget, J. *Adaptation Vtale et Pychologie de L'intelligence* (Adaptation and Intelligence: Organic Selection and Phenocopy), Translated by Stewart Eames, University of Chicago Press, Chicago, 1980.

Piaget, J. *Behavior and Evolution*, Translated by Donald Nicholson-Smith, Pantheon Books, New York, 1978.

Rist, R. "Schema Creation in Programming," *Cognitive Science* (13), 1989, pp. 389-414.

Rosson, M., and Alpert, S. R. "The Cognitive Consequences of Object-Oriented Design," *Human Computer Interaction* (5), 1990, pp. 345-379.

Rosson, M., and Carroll, J. "Climbing the Smalltalk Mountain," *ACM SIGCHI Bulletin* (21:3), 1990, pp. 76-79.

Rouse, W. B., and Morris, N. M. "On Looking Into the Black Box: Prospects and Limits in the Search for Mental Models," *Psychological Bulletin* (100:3), November 1986, pp. 349-363.

Rumelhart, D. E., and Norman, D. A. "Analogical Processes in Learning," in *Cognitive Skills and Their Acquisition,* J. R. Anderson (ed.), Lawrence Erlbaum Associates, Hillsdale, NJ, 1981.

Schenk, K. D., Vitalari, N. P., and Davis, K. S. "Differences between Novice and Expert Systems Analysts: What Do We Know and What Do We Do?," *Journal of Management Information Systems* (15), 1998, pp. 9-50.

Schmidt, R. A. *Motor Control and Learning: A Behavioral Emphasis,* Human Kinetics, Champaign, IL, 1988.

Sheetz, S. D., Irwin, G., Tegarden, D. P., Nelson, H. J., and Monarchi, D. E. "Exploring the Difficulties of Learning Object-Oriented Techniques," *Journal of Management Information Systems* (14:2) Fall 1997, pp. 103-131.

Shimp, T., Samiee, S., and Madden, T. "Countries and Their Products: A Cognitive Structure Perspective," *Journal of the Academy of Marketing Science* (21:40), Fall 1993, pp. 323-330.

Shneiderman, B. *Designing the User Interface: Strategies for Effective Human-Computer Interaction*, Addison-Wesley Publishers, Reading, MA, 1986.

Shneiderman, B. "Exploratory Experiments in Programmer Behavior," *International Journal of Computer and Information Sciences* (5:2), 1976, pp. 123-143.

Shneiderman, B., and Mayer, R. "Syntactic Semantic Interactions in Programmer Behavior: A Model and Experimental Results," *International Journal of Computer and Information Sciences* (7:3), June 1979, pp. 219-239.

Sircar, S., Nerur, S., and Mahapatra, R. "Revolution or Evolution? A Comparison of Object-Oriented and Structured Methods," *MIS Quarterly* (25:4), December 2001, pp. 457-471.

Soloway, E., and Ehrlich, K. "Empirical Studies of Programming Knowledge," *IEEE Transactions on Software Engineering* (10), 1984, pp. 595-609.

Spohrer, J. C., and Soloway, E. "Novice Mistakes: Are the Folk Wisdoms Correct?," *Communications of the ACM* (29:7), July 1986, pp. 624-632.

Stasz, C., Shavelson, R., Cox, D., and Moore, C. "Field-Independence and the Structuring of Knowledge in a Social Studies Minicourse," *Journal of Educational Psychology* (78), 1976, pp. 550-558.

Straub, D. W. "Validating Instruments in MIS Research," *MIS Quarterly* (13:2), June 1989, pp. 147-169.

Sumfleth, E. "Knowledge of Terms and Problem-Solving in Chemistry," *International Journal of Science Education* (10:1), January 1988, pp. 45-60.

Tapscott, D., and Caston, A. "The New Promise of Information Technology," *Ivey Business Journal* (57:4), Summer 1993, pp. 51-61.

Underwood, B. J. "Interference and Forgetting," *Psychological Review* (64), 1957, pp. 49-60.

Vessey, I., and Conger, S. A. "Requirements Specification: Learning Object, Process, and Data Methodologies," *Communications of the ACM* (37:5), May 1994, pp. 102-114.

Vitalari, N. P. "Knowledge as a Basis for Expertise in Systems Analysis: An Empirical Study," *MIS Quarterly* (9), 1985, pp. 221-240.

Weick, K. E. "Theory Construction as Disciplined Imagination," *Academy of Management Review*, (14:4), 1989, 518-531.

Wiedenbeck, S. "Novice/Expert Differences in Programming Skill," *International Journal of Man-Machine Studies* (23:4), October 1985, pp. 383-390.

Wiedenbeck, S. "Organization of Programming Knowledge of Novices and Experts," *Journal of the American Society for Information Science* (37:5) 1986, pp. 294-299.

Wittrock, M. C., and Cook, H. "Transfer of Prior Learning to Verbal Instruction," *American Education Research Journal* (12:2), 1975, pp. 147-156.

Yourdon, E., Whitehead, K., Thomman, J., Oppel, K., and Nevermann, P. *Mainstream Objects: An Analysis and Design Approach for Business,* Yourdon Press, Upper Saddle River, NJ, 1995.

About the Authors

Deborah J. Armstrong is an assistant professor of Information Systems in the College of Business at Florida State University. Dr. Armstrong's research interests cover a variety of issues at the intersection of IS personnel and mental models involving the human aspects of technology, change, learning, and cognition. Her research has appeared in *Journal of Management Information Systems, Communications of the ACM, Sex Roles,* and *The DATA BASE for Advances in Information Systems,* among others.

Bill C. Hardgrave is the Edwin and Karlee Bradberry Chair in Information Systems and Executive Director of the Information Technology Research Institute in the Walton College of Business at the University of Arkansas. His research on software development (primarily people and process issues) has appeared in *Journal of Management Information Systems*, *Communications of the ACM*, *IEEE Software, IEEE Transactions on Software Engineering, IEEE Transactions on Engineering Management, The DATA BASE for Advances in Information Systems*, *Information and Management,* and *Educational and Psychological Measurement*, among others.

Appendix A

Survey Instrument

Section I

A concept that is 0 percent *novel* is one that was originally defined in traditional development (any approach used that is not OO) and continues to hold the same meaning in OO development (borrowed concepts). A concept that is 100 percent *novel* is an OO concept that does not exist in traditional development. Concepts that are between 0 and 100 percent novel are concepts that have an existing meaning in traditional development, but now have a new and different meaning in the OO development context.

For example, if you know how to play tennis, but are learning to play racquetball, many of the concepts would be 0 percent novel. The concept of hitting a ball with a racquet is 0 percent novel. You hit the ball with a racquet in both tennis and racquetball. In contrast, the concept of hitting the ball off of a wall does not exist in tennis, so hitting a ball off the wall would be a 100 percent novel concept. The concept of keeping score would be between 0 and 100 percent because you keep score in both tennis and racquetball but you do it differently.

Please indicate the degree of novelty for each of the concepts listed below by placing an X on the line by the corresponding percentage.

Polymorphism	0% novel	10%	20%	30%	40%	50%	60%	70%	80%	90%	100% novel
Object	0% novel	10%	20%	30%	40%	50%	60%	70%	80%	90%	100% novel
Inheritance	0% novel	10%	20%	30%	40%	50%	60%	70%	80%	90%	100% novel
Class	0% novel	10%	20%	30%	40%	50%	60%	70%	80%	90%	100% novel
Encapsulation	0% novel	10%	20%	30%	40%	50%	60%	70%	80%	90%	100% novel
Attribute	0% novel	10%	20%	30%	40%	50%	60%	70%	80%	90%	100% novel
Method	0% novel	10%	20%	30%	40%	50%	60%	70%	80%	90%	100% novel
Message Passing	0% novel	10%	20%	30%	40%	50%	60%	70%	80%	90%	100% novel
Abstraction	0% novel	10%	20%	30%	40%	50%	60%	70%	80%	90%	100% novel

Section II

Please circle the letter corresponding to the best answer for each question.

1. If you extend a class you are
   a. Allocating more memory to the class
   b. Making the class more specific
   c. Creating extra instances of the class
   d. Creating a superclass

2. Abstraction is
   a. The process of creating an object from a class
   b. Cannot be modeled
   c. Adding values to an object's attributes
   d. Ignoring what is not important for the job at hand

3. In terms of object oriented programming, the use of polymorphism means
   a. That a client class does not need to be aware of the particular subclass that actually implements a method
   b. That objects change class over time
   c. That a single object is morphed into a number of database tables
   d. That subclasses can override the methods of a parent class

4. You model an object's characteristics by defining
   a. Methods
   b. Attributes
   c. Subclasses
   d. Procedures

5. __________ means that an object has attributes and methods combined into one unit.
   a. Inheritance
   b. Encapsulation
   c. Iteration
   d. Association

6. Every object must belong to at least one class.
   a. True
   b. False

7. In object-oriented systems inheritance
   a. Shows how a class changes over time
   b. Shows how messages are passed between classes
   c. Is only a programming technique to reduce the amount of code
   d. Allows similarities and dissimilarities to be modeled clearly

8. __________ refers to the way different object can respond in their own way to the same message.
   a. Encapsulation
   b. Inheritance
   c. Polymorphism
   d. Association

9. A class is
   a. Different than an object because a class is an instance and an object is a category
   b. The same as an object
   c. Different than an object because a class is like an instance and an object is like an association
   d. Different than an object because a class is a type of thing and an object is an instance of a thing

10. The process by which an object sends information to another object or asks the other object to invoke a method is known as __________.
    a. Method processing
    b. Abstraction
    c. Instantiation
    d. Extension

11. A class that allows a user of that class to access its variables *only* through the use of the class's methods has incorporated the object-oriented principle of
    a. Polymorphism
    b. Encapsulation
    c. Inheritance
    d. Extension

12. An object's ability to decide which method to apply to itself depending on where it is in the hierarchy is called
    a. Encapsulation
    b. Cloning
    c. Polymorphism
    d. Early binding

13. Every class must have at least one object.
    a. True
    b. False

14. A method may return at most
    a. 0 values
    b. 1 value
    c. 2 values
    d. Any number of values

15. A class specifies the behavior of its instances.
    a. True
    b. False

16. Objects are aware of how other objects are implemented.
    a. True
    b. False

17. With regard to an auto, the driver doesn't care how the engine runs, the mechanic doesn't care about the inner workings of the battery, but the battery manufacturer does. This is an example of
    a. Polymorphism
    b. Message Passing
    c. Method
    d. Abstraction

18. Objects interact by sending messages to one another.
    a. True
    b. False

19. Methods are always associated with a specific object.
    a. True
    b. False

20. There is no way for an object to access the data of methods in another object.
    a. True
    b. False

21. Each object of a class will have the same values for its attributes.
    a. True
    b. False

22. If the Rental Equipment class is responsible for the processing required when it "rents itself," then the processing would be a(n)
    - a. message
    - b. method
    - c. object
    - d. class

23. You want to move an automated vehicle (AV104) to a new location (binB7). To accomplish this, you would
    - a. Send a message to vehicle AV104 and ask it to move to binB7
    - b. Send a method to vehicle AV104 and ask it to move to binB7
    - c. Have the AV class relocate to binB7
    - d. Change the attribute of the vehicle to show the new location as binB7

24. For an object named Product, which of the following is a likely attribute?
    - a. Description
    - b. SetDescription
    - c. AddToOrder
    - d. GetPrice

25. Abstraction is the process of focusing on those features that are essential for the task at hand and ignoring those that are not.
    - a. True
    - b. False

26. According to the concept of __________, one class of objects can take on characteristics of another class and extend them.
    - a. Encapsulation
    - b. Lineage
    - c. Association
    - d. Inheritance

27. Which statement is true concerning objects?
    - a. Objects are a collection of competing things that can be classified as a specific type
    - b. Objects are typically complex
    - c. Objects cannot be reused in other systems
    - d. Objects are self-contained

Section III

Think back to when you first learned the concepts listed below. Please indicate how easy or difficult it was for you to learn each concept by circling the number that indicates the level of learning difficulty.

	Very Easy	Moderately Easy	Slightly Easy	Neutral	Slightly Difficult	Moderately Difficult	Very Difficult
Object	1	2	3	4	5	6	7
Message Passing	1	2	3	4	5	6	7
Abstraction	1	2	3	4	5	6	7
Method	1	2	3	4	5	6	7
Polymorphism	1	2	3	4	5	6	7
Attribute	1	2	3	4	5	6	7
Inheritance	1	2	3	4	5	6	7
Encapsulation	1	2	3	4	5	6	7
Class	1	2	3	4	5	6	7