Applying Bloom's and Kolb's Theories To Teaching Systems Analysis & Design

Laurie Schatzberg
Anderson Schools of Management
University of New Mexico
Albuquerque, NM, USA 87131-1221

ABSTRACT

Enabling students to understand the processes of systems analysis and design is a challenge facing all MIS educators. While the concepts and details are covered thoroughly in texts, it remains difficult to convey the big picture of the development process. Due in part to the inherent complexity of the subject and the relative absence of introductory-level experiential exercises, beginning MIS students often struggle to understand concepts well enough to apply them in later courses and in the field.

Learning theorists such as Bloom and Kolb, however, underscore the importance of experiential learning to developing mastery of a subject. The current work applies these theories to teaching information systems analysis and design in general, and then presents a Lego-based classroom activity to begin to fill the need for early experiential learning in an introductory analysis and design course. Whether students are beginning to learn the field or already have some background, the Lego-based activity is a rich metaphor for the entire systems development process. Students can use the lessons from this activity as they progress through the course and beyond.

The activity is one of a growing suite of similar Lego-based activities that has been used for five years at the University of New Mexico in such courses as Introduction to MIS, Structured Systems Analysis & Design, and Object-oriented Analysis & Design.

Keywords: Introductory experiential learning; learning theories; analysis & design

1. INTRODUCTION

Systems development is an integral component of an MIS curriculum. One way to characterize the goal of MIS curricula is that we seek to educate students to be technology generalists with a thorough knowledge of business. Nearly ten years ago, Mary Boone (1993) coined the phrase "boundary person" to characterize these skills – that is, students who can communicate and work across the boundaries separating IT from the rest of an organization. We seek to create graduates who are instrumental in developing technology-based systems that directly support organizational goals. Thus, our students must be well schooled not only in technical skills but also process skills, such as those defining systems development, such as analysis and design.

Notable model curricula at both undergraduate and graduate levels (Davis, 1997; OSRA, 1996; and Cohen, 2000) require that all management students at least develop MIS literacy and that majors should develop greater competence. Clearly, while analysis and design concepts and skills have direct application to careers in MIS, they are also relevant general problem solving, project management, systems development, diffusion of innovation and organizational change work.

In particular, IS97 cites System Analysis and Design as one of the eight (8) core skill categories needed by MIS professionals (Davis, et al., 1997). Similarly, the OSRA model (1996) calls for mastery of systems development concepts (see OEIS-1; OEIS-3 and OEIS-4) beyond the basic level. The IRMA/DAMA model (Cohen, 2000), focusing primarily on the courses needed to develop information resources managers also devotes considerable focus on systems analysis and design (see IRM6 in Cohen, 2000). The graduate curriculum, found is MSIS2000 (Gorgone & Gray (eds.), 1999) calls for similar focus on these skills, both in the IS core for graduate programs and in the electives.

Academic and practitioner communities seem to agree that management and, in particular, MIS students should become well versed in analysis and design concepts and skills. Thus, it is reasonable to create courses that maximize both mastery and retention of the skills and concepts. The current work approaches this goal by applying learning theories to MIS pedagogy. In particular, this work focuses on strengthening introductory systems development courses, where the focus is upon information systems analysis and design and associated modeling methodologies. The theoretical foundation for this work is Bloom (1956) and Kolb (1984).

In the theory section that follows, Bloom's taxonomy of educational objectives is presented and applied to introductory systems analysis and design course. Then, Kolb's experiential learning theories are presented to suggest strategies for conveying systems development skills to MIS students.

Following the theory sections is a section focusing on a classroom Lego™ experiential exercise. This exercise is one of a suite of class activities aimed at moving students more consistently to higher levels on Bloom's taxonomy. Anecdotal results are presented. In the final sections, we highlight strengths and weaknesses inherent in this work, suggest ways to quantify the results and, ultimately, assess students' learning in an information systems development course.

2. TWO UNDERLYING THEORIES

Two learning theories provide the foundation for this work. First, is the work by Bloom and others (1956) whose work on cognitive operations resulted in a taxonomy to classify the extent to which an individual understands a given concept. This taxonomy is still widely accepted and in use today (Kottke & Schuster, 1990;Granello, 2001). The taxonomy identifies six hierarchical levels of learning: Knowledge, Comprehension, Application, Analysis, Synthesis, and Evaluation.

The second theoretical underpinning is derived from Kolb (1984) whose work suggests that experience is a crucial mechanism to impart concepts that can subsequently be used for extrapolation and active experimentation.

In the sections that follow, we introduce each of these theories and apply them to the current context: a course in systems analysis and design. This type of course is offered only to MIS majors or minors and has as prerequisites the introductory survey-of-MIS course and at least one programming language.

Applying Bloom's Taxonomy to Systems Analysis & Design

The six levels in Bloom's (1956) taxonomy are Knowledge, Comprehension, Application, Analysis, Synthesis, and Evaluation. These levels are assumed to be cumulative, with each level building on the successful integration of the previous levels. Much research has been conducted on the model, and it has been found to transcend age, type of instruction, and across fields such as nursing, computers, economics, sociology, accounting, and physical therapy (Hill & McGraw, 1981; Kunen & Solomon, 1981; Kottke & Schuster, 1990; Niehoff & Whitney-Bammelin, 1995). In an investigation that is similar to the current work, Kraiger & Salas (1993) used Bloom's taxonomy to evaluate learning outcomes in organizational training programs.

The following sections explain each of the levels and apply them to learning IS analysis and design.

Knowledge At this initial learning level, the student recalls or recognizes information, ideas, and principles in the approximate form in which they were learned. The material may vary from specific facts to process steps to theories, but students remember the information. With this level of learning, students can recite information without demonstrating any understanding. They can recall factual information and can recognize examples that fit (or don't fit) a given concept. This level of learning can be demonstrated.

At the *knowledge* level, for example, students are able to state the phases of information systems development, because they have memorized definitions and/or lists of tasks. They can pick out the steps of systems development from a list, and can order them chronologically. With respect to information modeling, they can recognize types of models they have been shown, such as a data flow diagram (DFD) or a class diagram (CLD). They recognize these models easily when the new model uses the same symbol set they saw before. Since modeling techniques vary considerably in the structured methodology and even within object-oriented (UML-based) methodology, students require higher levels of learning to recognize and work with varying styles of information modeling.

At this *knowledge* level, students are not be able to identify or summarize main points of a system narrative, to distinguish high quality (appropriate) from low quality (inappropriate) development approaches. They are not able to interpret a DFD or CLD, to recognize modeling errors or omissions, nor to construct their own DFD or CLD from a narrative or observation.

Comprehension The *comprehension* learning level is the ability to grasp the meaning of material and can be demonstrated by translating material from one form to another or by interpreting material. Evidence of this stage is one's ability to identify and then articulate in one's own words the main concepts.

In the MIS context, students demonstrating *com-prehension* can summarize and translate the main points of the development process using their own words. They can explain the roles of MIS professionals, end-users, and managers. With respect to modeling system requirements, students can explain the symbols on a DFD or CLD and can interpret such a model. They can explain (translate) an information model (e.g., DFD or CLD) to an end-user. They cannot detect modeling errors or omissions and cannot construct their own model from a narrative or an observation.

Application is the ability to use learned material in new and concrete or specific situations. *Application* involves extending and predicting how concepts work in a new setting, and includes applying rules, methods, concepts, principles, and/or theories that have been mastered from lower learning levels.

In systems modeling, students can create an accurate DFD or CLD based on a prepared narrative – where the key facts are given and extraneous information is removed. They can also create a narrative based on an observation of a systems development activity. At the *application* learning level, however, students cannot complete a DFD or CLD model based on their own observations or based on a free-form narrative, since they cannot yet distinguish significant details from unimportant ones (Granello, 2001). Moreover, their own narratives may be filled with irrelevant details and may lack crucial ones – because, again, students at the *application* level cannot yet decide for themselves what is relevant.

Analysis refers to the ability to break down material into its component parts and includes the identification of parts (of a whole), discovering the relationships among the parts, and recognizing their organizing principles. At this level, MIS students develop the ability to draw out significant information (that is, requirements) from users. They are able to separate minutiae from relevant facts in users' explanations. At this level, they begin to organize what they hear and read from users as these things might impact their subsequent development work.

They begin to understand, for example, that a given requirement for, say, "ad hoc reporting of sales data" creates specific technical requirements for hardware, software, user interfaces and training. They will also understand that it is not particularly important that a report is currently given to exactly seven sales reps; it is important, however, that the report is available to multiple individuals whose role is sales rep. At this level, students can choose the correct information models (e.g., DFD, ERD, CLD, or other OO models) to capture the requirements. They recognize inconsistencies or anomalies among users' requirements and can detect both syntactical modeling errors (incorrect symbols and/or associations among symbols) and semantic modeling errors (incorrect transformation of narrative/observation into model or specifications).

This level of skill already has applicability to the MIS profession, and students at this level of learning will be positive contributors to live systems development work. This *analysis* level of mastery is the goal of the IS course discussed in this work. The remaining two levels begin in this course, but mastery of these levels is only achieved later in the curriculum, at which time students have been engaged in a greater variety of learning experiences and over an extended period of time.

Synthesis refers to the ability to put parts together to form a new whole and to combine pieces into a pattern or system that didn't previously exist. The student originates, integrates, and combines ideas into a product, plan, or proposal that is new to him or her. At the *synthesis* learning level, students are able to suggest and develop meaningful alternatives to solve a given problem. For example, they understand the user requirements and constraints enough to propose reasonable alternatives to users' stated problems and concerns. That is, they understand the logical system requirements well enough to generate alternative physical solutions.

Synthesis differs from the *analysis* level in that students are able to move beyond what is given to them (e.g., an "as is" system) and to conceive the "to be." They can organize and relate inputs from users, theories, principles and concepts, and previous experiences in order to define a new system. Further, they can use their own requirements analysis as the basis for information models and can iterate between modeling and requirements gathering to fill in gaps and resolve inconsistencies. They are able to interpret written documents and observations and to integrate any meaningful content into new system requirements. Solid undergraduate MIS programs produce students at this level of learning.

Evaluation, the highest level of learning in this taxonomy, is ability to judge the value of material for a given purpose. Such judgments are based on defined criteria that are either developed by the individual or provided by outside sources (professional standards, instructors, texts, colleagues). *Evaluation* contains elements of all the other categories and involves conscious, independent value judgments based on clearly defined criteria.

Students at this level can critique and judge the quality of material they are given, by applying rules, heuristics, and criteria that they have amassed for this purpose. In systems development work, for example, students can critique and compare different models representing a single set of user requirements. They can critique the models based on such established criteria as (1) technical/syntactical correctness of the model and (2) accuracy with which the explicit and implied requirements are captured.

This level of mastery does not suggest that the individual performs perfectly, but rather that they have the ability to evaluate according to agreed upon standards. This level also suggests that the end product of their work will meet the required standard of performance.

Individuals at the *Evaluation* level also recognize that differing assumptions and methodologies can yield differing and sometimes-contradictory results. They distinguish differences in "style" from errors. They can resolve the discrepancies or accept inevitable ambiguity. This skill level is the intended outcome of graduate MIS curricula.

In this section, we outlined Bloom's taxonomy and applied it to course content in information systems development. In the next, we outline and apply Kolb's work to suggest techniques that enable students to move steadily through these levels of learning.

Applying Kolb's Learning Model to Systems Analysis & Design

Kolb's (Kolb, 1984) four-stage model presents a learning cycle to depict how *experience* is translated through *reflection* into *concepts*. These concepts are then used to guide active experimentation and the choice of new experiences. These four stages derive from the two major ways by which individuals learn: (1) perceiving or grasping new information or experience, and (2) integrating or transforming what is perceived (Smith and Kolb 1986) into concepts. Kolb (1984) refers to these four stages as concrete experience (CE), reflective observation (RO), abstract conceptualization (AC), and active experimentation (AE). They follow each other in a cycle and provide feedback, enabling evaluation of previous actions and plans for new actions.

Movement through these stages is iterative and the cycle unfolds in the sequence shown in the figure below, adapted from Kolb (1984, p.42). The dynamic nature of interactive learning cycles represents a spiral of learning cycles (Healy, 2000). While a cycle can begin at any stage, a common learning cycle begins with a concrete experience, after which the individual engages in some reflection of that experience in order to understand it and integrate it with other, prior knowledge. Abstract conceptualization follows if the individual attempts to extrapolate understanding beyond his/her concrete perceptions. Finally, an individual will begin to apply the learning in new, experimental fashion, testing the concepts, refining their understanding, observing results, integrating feedback and perhaps cycling through another time to develop deeper learning.

Examples of the *grasping* dimension are immersion in a concrete activity such as a hands-on task (concrete experience) and, in contrast, thinking abstractly such as interpreting written material (abstract conceptualization). Similarly, examples of the *transforming* dimension are doing a new activity (active experimentation) and watching (reflective observation) (Fielding 1994). While not the topic of the present work, these dimensions are also used to classify learning styles and professional characteristic.

The application of Kolb's work to the present context involves identifying class activities and assignments that engage students in each of the stages of learning – thereby ensuring that, regardless of students' learning styles and pace, class content will reach all students. Further, by constructing a variety of experiences that are explicitly aimed to foster grasping and transformation, students heighten their understanding of the material and move steadily from *Knowledge* to *Comprehension* to *Application* to *Analysis* learning levels.

Section 3 describes an exercise that incorporates a full learning cycle. Section 4 maps the exercise to the learning theories.

3. LEGO-BASED EXPERIENTIAL EXERCISE

This section describes one exercise (prototyping) from a suite of Lego-based exercises that illustrate and apply the learning theories discussed. The suite of exercises provides an inexpensive and concrete way to improve students' understanding of systems analysis and design early in their MIS coursework.

Much like an engineering or science discipline, working in systems development is quite different from simply learning the concepts and skills involved. The discipline requires the ability to integrate technical knowledge with interpersonal and communication skills. The experiential exercise aims to impart a sense of this systems development gestalt.

Incorporating these exercises early in the curriculum is not intended to substitute for the later activities, nor is it intended that Lego-based activities are the sole pedagogical technique for a course. Instead, positioning this experiential exercise early in the MIS students' program is intended to enhance their learning and retention. These exercises complement other activities within the systems analysis and design course, and fill a common gap.

These exercises help students move from the *knowledge* level of mastery through to an *analysis* level, as they learn the concepts, their distinctions and relationships through simple analogies. Since the analogies can be understood quickly, students can soon step into the learning cycles for true systems development.

In particular, each exercise engages students in all phases of Kolb's learning cycle and prepares them to move through subsequent learning cycles as they progress through the course and the curriculum (i.e., through levels in Bloom's taxonomy).

Each activity in this suite of experiential exercises uses simple Lego™ blocks to create the metaphor for information systems development. For all activities in the suite, students work in small teams. These teams can be self-selected or appointed. Each activity can be completed in a single 60 to 75-minute class session and includes hands-on action activities, observation and reflection, as well as discussion. The activities are straightforward and yet they are rich analogies for the actual work involved in complex systems development. Debriefing each exercise through reflection and discussion draws out the metaphors and is key to the success in each activity.

Metaphors range from resource constraints (e.g., time, skills, materials & supplies), team management (e.g., leadership, planning, negotiating), project management (e.g., meeting deadlines, interacting with the customer, overcoming obstacles, meeting requirements), interpersonal communications (e.g., interviewing and presentation) as well as analysis and logical design concepts (e.g., identifying requirements, scoping the project, identifying alternatives, choosing among alternatives, implementing the chosen design, documenting the design and process, implementing the system, maintaining the system and/or customer support).

The debriefing then focuses on what they did, how they did it, difficulties, perceptions, and reinterprets them all from perspective of real systems development. Almost uniformly, students retain the lessons from these activities. They demonstrate their learning on quizzes, tests and projects, and in subsequent courses.

4. EXAMPLE EXERCISE: PROTOTYPING & IMPLEMENTING A NEW SYSTEM

Exercise
Create an Executive Desk Toy for your customer.

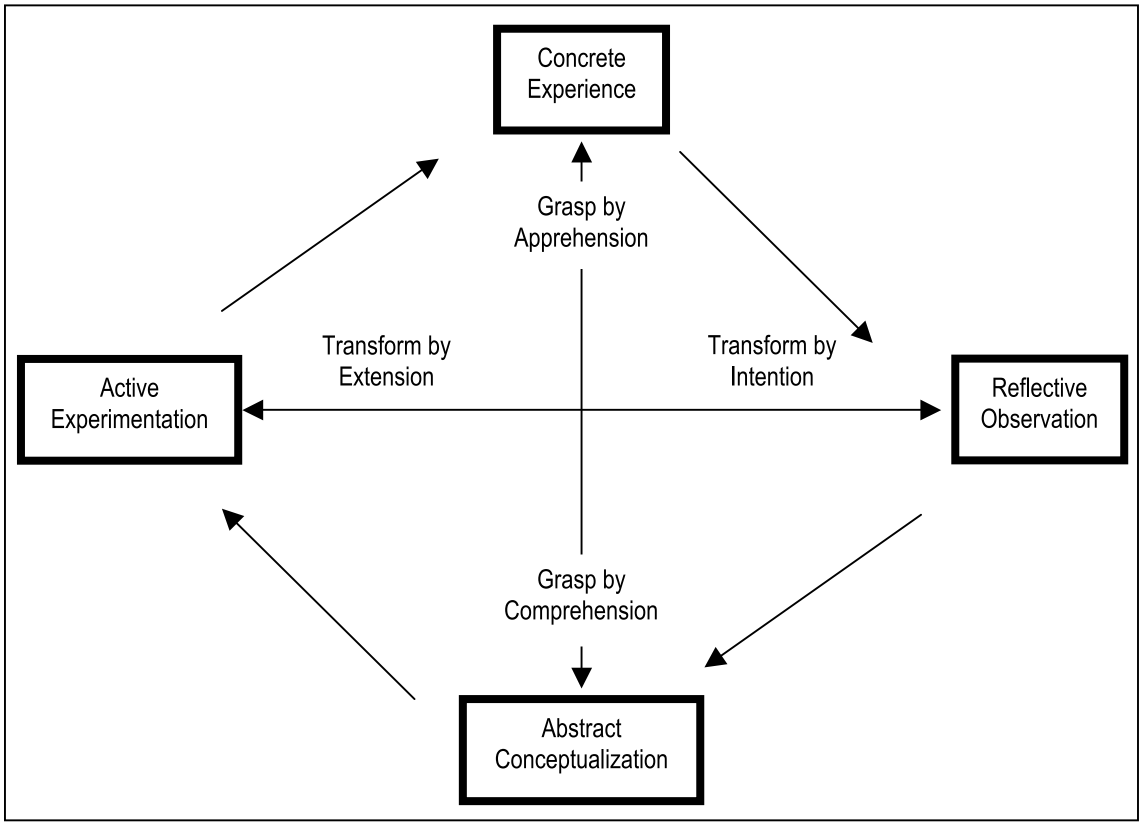
Context
The Development tasks in this exercise are a metaphor for system development using a modified prototyping approach, and the Assembly tasks are a metaphor for implementing the prototype as a final system.

Materials
Create packages with about 20 Lego pieces each; small zipper-type bags work well for packaging. Before class, place a variety of Lego pieces, including some interesting/movable pieces into each package. The assortment can vary across packages. The intent is to provide pieces that stimulate creativity and suit the task. Label each bag with a number to identify the team who will use it for Development. Create more packages than teams, so that each team has a choice of packages. Include 1-2 sheets of paper or index cards in each package also.

Setup
In class, explain the entire exercise (see Appendix for URL to course material). Organize students into teams of 3-5 students. These are temporary teams who work together only for this exercise. In classes where students complete semester-long projects in teams, it is useful to conduct this exercise at about the time that the more-permanent teams are forming. By doing so, the exercise becomes part of the students' foundation to enter a subsequent learning cycle of systems development in the Active Experimentation stage.

The Process
The exercise is comprised of two main sets of tasks: Development (20 minutes) and Assembly (10 minutes). When those tasks are completed, the instructor guides the class through Evaluation (10+ minutes) and Debriefing (20+ minutes). While the exercise refers to Development teams and Assembly teams, all teams engage in both activities.

Development Part A: The Tasks
Each team selects a package from the instructor's inventory. During this part, teams (1) create a Lego product using at least seven pieces, (2) name their product and write that name on the paper provided, and (3) write assembly instructions for their product. Later, the Assembly team will use the Development team's



instructions to construct the Executive Toy. Remind the students to include their Team Number on the assembly instructions.

If there are unused packages and/or additional Lego inventory, make this inventory available for teams to supplement their own working inventory. Teams will often want one more wheel, person, or specific-colored/-sized piece complete their design. Leave the inventory in a central location where all teams can access it.

Development Part B
Instruct all the teams to *completely* disassemble their product and return all their Lego pieces (whether or not the pieces were used in their product) to their zipper bag. Instruct the teams to place their assembly instructions in their bag, close the bag and return the whole package to the instructor.

Assembly Part A
The instructor gives each Assembly team a package containing work from another Development team. Assembly teams use *only* the instructions contained in the package to complete their work. A variation allows the Development team to serve as consultants to the process.

Assembly Part B
The Assembly teams should reflect upon their assembly process, noting the things that made their work easy or difficult. As each assembly team is done, they bring their finished product and assembly instructions to the display area in the classroom. They return the unused materials to central inventory. Finally, each Development team inspects the work of their Assembly team.

Evaluation
Led by the instructor, the class participates as a whole in evaluation. Call upon each Development team to report their evaluation of the assembly work done by "their" Assembly team. Similarly, call upon the corresponding Assembly team to evaluate the quality of the instructions they received from the Development team.

This evaluation and discussion can take 10-15 minutes, depending on the number of teams and the depth of the commentary each team offers. Further, the evaluation segment is tightly coupled with the Debriefing segment described below. Explicitly integrate these two segments to more easily draw out system development analogies from students.

Debriefing
When asked to reflect upon the evaluation information, teams often see parallels and analogies to MIS work they have experienced or read about. Drawing out those comparisons is the purpose of debriefing. The instructor should draw as much as possible from the students and reinforce the correct points they make. The instructor should also be prepared to fill in what the students may miss, and to respond to unexpected outcomes. Commonly, debriefing uncovers the following:

Resource constraints Development packages/ Lego inventory is insufficient; drawing/documenting skills of Development team members is limited; insufficient time to complete Development or Assembly tasks.

Customer interactions In this exercise there are two customers: the instructor who provided the original requirements for the prototype, and the Assembly team who is the end-user for the final products. The parallels to customer issues in systems work include vague/insufficient requirement inputs to Development team from [instructor] customer; unskilled/inept [Assembly team] customer; customer satisfaction (or not), insufficient or unclear support to [Assembly team] customer by the Development team.

Team performance: Highly co-operative teamwork; task-focus; role specialization within team; pride in work and accomplishment (surprisingly, teams become "attached" to their product and are a bit unhappy to have to disassemble them when class is over).

Development process A simple design that meets the customer's requirement (not much more than 7 pieces) is easier to create and easier for customers to assemble than a complex design involving dozens of parts; Creating documentation for *someone else to use* is difficult, time-consuming, and not as much "fun" as creating the product itself; instructions created by others are unclear and incomplete; documentation is good when it is succinct and includes pictures; delighting the customer and meeting the deadline pulls the team in contradictory directions, prototyping resulted in much more rapid results for the customer than a more formal system development process; when team members take on responsibility for differing roles and accept input from other team members, the result is high quality work and enhanced trust among members.

Instructor Involvement During the Exercise
While teams are working in parallel on Development and Assembly, the instructor has the opportunity to observe them, clarify any customer requirements, and note additional parallels to systems development work. Most, but not all teams will meet the 20-minute deadline for their Development work. To achieve maximal success on this dimension, it is useful to provide signals and reminders as the time passes. When 20 minutes has elapsed, however, it is important to stop the class to recognize the teams (vendor analogy) who met the customer's deadline and to reward them with some token. In addition, this group of teams should be allowed to begin the Assembly work, and should not be required to wait until everyone else is completed. It is useful to note the extra time that slower teams use. During debriefing, analogies can then be drawn to meeting customer deadlines. For example, 5 minutes late in a 20-minute activity is a missed deadline of 25%. In real systems development work, that magnitude of slipped schedule can result in the vendor losing, instead of making, money on their contract.

As the analogies are discussed, students may point out that “assembly kits” don’t normally contain extraneous parts, while this exercise required them to return all their original parts to the assembly package. While this observation is correct for physical products, it is not correct for software or information-based products. Most such products ship with a set of files that are used in some, but not all, installations; for example, drivers needed for specific operating systems. Vendors and teams must diligent to ensure that users install the right components and ignore the irrelevant (to them) files and data. Further, instructions must enable users to recognize and recover when they make errors such as pressing the wrong keys, or entering incorrect data into a field. These user situations are another direct analogy to the “extra parts” feature in this exercise.

5. MAPPING THE LEGO EXERCISE TO BLOOM’S AND KOLB’S THEORIES

Because students take on different roles during the exercise, and because classroom dynamics will alter the exact nature of the exercise, students will move through the learning cycle somewhat differently. Yet it is straightforward to map their learning to the learning theories discussed previously.

Here are typical mappings to a learning cycle for prototyping information systems. Refer to the figure above to follow the cycle.

Active Experimentation
Students approach the activity with fairly well developed “Lego-skill” and they have learned how to attach pieces just for the joy of building or to complete a Lego kit. Typically, they have also seen desktop toys and understand their purpose (relaxation or conversation starter) and size (small). In this exercise, they engage in active experimentation as they structure the Development task for themselves. Purposefully, there are few instructions give to them.

Concrete Experience
The team quickly begins to prototype a toy, and as they do so, students are engaged in the concrete experience of using Lego to create a desktop toy.

Reflective Observation
Observing others’ participation (the complementing roles to their own); recognizing what worked and what didn’t work within their team or for other teams. In this stage, they focus on the Lego exercise itself, however.

Abstract Conceptualization
This stage is reached when students grasp the parallels to real systems development work. They extrapolate at a conceptual level to understand team dynamics, customer interactions, resource management and related issues.

Turning to Bloom’s levels of learning, students who begin the exercise with rote *Understanding* level of learning, will move to at least to *Comprehension* at the end of this exercise. The evidence is that they will be able to explain in their own words many of the concepts represented in the exercise. Those who begin the exercise at the *Comprehension* level will move at least to *Application* because they will have applied the system development concepts to the Lego exercise. Students demonstrate this achievement in their team (concrete experience) and may also describe it in the Debriefing. Students who identify systems prototyping analogies during the Debriefing are demonstrating *Analysis* level of learning.

6. CONCLUSIONS, LIMITATIONS, & FUTURE WORK

Teaching systems analysis and design to beginning MIS students is both essential and challenging. While such classes invariably include CASE-based or paper-based activities, they often lack experiential components that allow students to deal with prototyping from start to finish.

The theories offered by Bloom and Kolb are useful frameworks within which to develop experiential learning activities in systems analysis and design classes. The Lego-based exercise seems to provide meaningful analogies to convey complex systems development concepts. This exercise, used as one of several pedagogical techniques in the classroom, can move students quickly through multiple levels of learning.

While the exercise has been used for a number of years at the University of New Mexico, evidence of its usefulness remains anecdotal. It is important to evaluate the usefulness of this type of exercise more rigorously.

REFERENCES

Bloom, B. S., Engelhart, M. D., Furst, F. J., Hill, W. H., & Krathwohl, D. R. (1956). Taxonomy of Educational Objectives: Cognitive Domain. New York, McKay/Longmans Green.

Boone, M. (1991). Leadership and the computer. Roseville, CA: Prima Publishing.

Cohen, Eli (ed.) “Curriculum Model 2000 of the Information Resource Management Association (IRMA) and the Data Administration Managers Association (DAMA),” accessed 5/2002 from **http://gise.org/IRMA-DAMA-2000.pdf**

Davis, G. B., Gorgone, J. T, Couger, J. D., Feinstein, D. L., and Longnecker, Jr., H. E. (Eds.) (1997). "S'97 Model Curriculum and Guidelines for Undergraduate Degree Programs in Information Systems." The Data Base for Advances In Information Systems **28**(1).

Gorgone, J. and Gray, P. (eds.) ,“MSIS 2000 Curriculum 2000” *Database for Advances in Information Systems*, Vol. 31:1, Winter 2000.

Granello, D. H. (2001). "Promoting cognitive complexity in graduate written work: using Bloom's taxonomy as a pedagogical tool to improve literature reveiws." Counselor Education & Supervision **40**(4): 292-307.

Healy, M. and Jenkins, Alan (2000). "Kolb's experiential learning theory and its application in geography in higher education." Journal of Geography **99**(5): 185-195.

Hill, P. W., and McGaw, B. (1981). "Hill, P. W., & McGaw, B." American Educational Research Journal **18**: 92-101.

Kolb, D. A. (1984). Experiential Learning: Experience as the Source of Learning & Development. Englewood Cliffs, NJ, Prentice-Hall.

Kottke, J. L., & Schuster, D. H. (1990). "Developing tests for measuring Bloom's learning outcomes." Psychological reports **66**: 27-32.

Kraiger, K. F., J. K., and Salas, E. (1993). "Application of Cognitive, Skill-based, and Affetive Theories of Learning Outcomes to New Methods of Training Evaluation." Journal of Applied Psychology **78**: 311-328.

Kunen, S., Cohen, R., & Solmon, R. (1981). "A Levels-of-Processing Analysis of Bloom's Taxonomy." Journal of Educational Psychology **73**: 202-211.

Longnecker, H., Davis, G., Feinstein, D., Gorgone, J., Valacich, J., "IS 2001: Updating IS'97, A Progress Report on Undergraduate IS Curriculum Development," presented at AIS Conference, Boston, MA, accessed 5/2002 from **http://www.is2000.org/AIS2001.ppt**

Meyer, N. D., & Boone, M. (1987). The information edge. Homewood, IL: Business One.

Niehoff, B. P., Whitney-Bammelin, Donita L (1995). "Don't let your training process derail your journey to total quality management." Advanced Management Journal **60**(1): 39-xx.

OSRA, "Organizational and End-user Information Systems," (OEIS –1; OEIS –2, OEIS –3 and OEIS –4) Office of Systems Research Association, 1996, accessed 5/2002 from http://home.nyu.edu/~bno1/osra/model_cur-riculum/OEISINTRO.html

APPENDIX

To Access Instructions & Related Material for Students:
ftp://ftp.mgt.unm.edu/LaurieSchatzberg/459/