Experiential Learning Theory and the Teaching of Software Intensive Courses

A. Richard Vannozzi, MS, PLS
Thompson School of Applied Science, University of New Hampshire

Session: Tools, techniques, and best practices of engineering education for the digital generation

Each year, pressure seems to grow to bring more software into our teaching. There are three forces at work in engineering education today that make a discussion of how to teach software intensive courses timely. First, initiatives such as laptops in the classroom, asynchronous and blended learning models, Web 2.0 and the multi-media classroom have all brought fundamental changes to the modern engineering learning environment. Second, the students who come to us, for the most part, are technologically savvy and have a comfort with utilizing technology in their education that often outstrips our own. Third, engineering software is the workhorse in the modern engineering practice and research environments we are preparing our students for. Software crunches our numbers, brings our ideas to life, and even controls the manufacturing and construction that springs from engineering designs. If anyone is still wondering, software is here to stay and debate no longer can center on whether we utilize software in our teaching, but rather how do we utilize software effectively in order to best help our students achieve the learning outcomes we define.

To say that software centered pedagogy has replaced, or should replace other time tested pedagogies would be as naïve as it would be foolish. In the same way that not all tasks a practicing engineer does in their workday are centered on software, not all learning is best accomplished using a software platform. Similarly, some tasks are so software intensive that to try and teach them without software would be ludicrous. This paper focuses on two such specific aspects of engineering education where software is essential. The first is, by its name and nature software intensive, that being Computer Aided Design (CAD) and the second is Geographic Information Systems (GIS), an example of the database driven modeling environments that are growing in use in nearly all domains of engineering.

Because software to accomplish specific functions is developed by differing and competing vendors, and even software by a particular vendor changes over time, approaches to educating engineering students fails if all that is accomplished is the creation of automatons who can memorize a series of button pushing sequences that provide a pleasing outcome. Focusing on a specific button pushing sequence to reach a desired outcome is often referred to as "teaching the tool" or "teaching to the tool". Because the number of button pushing sequences is nearly infinite and dynamic, it is really the decision about what needs to be accomplished that is of primary importance and then, secondarily developing in our students the reasoning skills needed to find a useful button. This involves understanding what one needs the buttons to do before one begins pushing them. Only then can the user thoughtfully choose which button from the specific software's array to push and then be able to assess whether the result of the button pushing accomplished the technical goal. This is akin to having a hypothesis about what you expect the result of a scientific experiment to be before you embark on your lab work. And just like scientific lab work, CAD or GIS learning is not complete without both the hypothesis and the experiment.

Before discussing specific methodologies for teaching and learning in software intensive environments, it is important to draw a distinction between training and education. Training is usually understood to be the acquisition of a specific set of skills in order to master the performance of a specific task, with little emphasis on theory. [1]. This is not to say that theory is not important, and in fact implicit in most professional training is the fact that an understanding of the underlying theory and abstractions already exists in the trainee. In contrast, education is characterized by the transmission of knowledge and understanding for use in subsequent dynamic applications [1]. Implicit in education is the exposure to theory as at least part of the learning experience. Step by step button pushing lessons in order to understand how specific software accomplishes some task that is already being accomplished by another method or software has a viable place in a training environment. If an individual already knows how and why to do something, but just needs to harness a specific piece of software to do it, that is a very different situation from one where the theoretical abstractions or hypotheses are not known or understood to be true, as is generally the case in educating undergraduate engineering students. Most software manuals, on-line help and many tutorials are developed for practitioner training purposes. Since it is industry that pays the vast majority of the software acquisition costs, (not the educational sector) it should not come as a surprise that the materials made available from the vendors are designed to meet the needs of the industry customer where it is reasonable to assume that the abstractions are already understood. Though some CAD and GIS educational materials are evolving from training materials they still have as a primary component "button pushing sequences". Unfortunately both students and educators often gravitate away from an educational mode to a training mode and abandon the abstractions and find themselves "teaching to the tool".

Experiential Learning Theory (ELT) as posited by D. A. Kolb in his 1984 book: *Experiential Learning; Experience as the Source of Learning and Development* [2] provides a single coherent framework that explains both the process of learning and how learning is experience based. By adhering to ELT in instructional design for CAD, GIS and other software intensive courses, classroom approaches can be crafted to keep the software in its proper place as a tool. ELT provides a unified context for understanding the relationship between the abstract "hypothesis" and the concrete result of the button pushing. Kolb's conceptualization of these is shown in Figure 1. Zull refers to the components of Kolb's ELT as the "four fundamental pillars of education: gathering, reflecting, creating, and testing. [3]"

Irrespective of which instructional design model or process one might prefer, at some point learning objectives are defined and then a specific activity is undertaken in order to achieve that objective. The activity could be to read a book, listen to a lecture, do a homework problem, perform a lab experiment, make a movie, write a paper, etc. These activities could form the assessment of the learning, or a separate follow up assessment might be involved, such as an exam. But the essential elements of objectives, learning and assessment can be found at the heart of most mainstream instructional design models (e.g. Dick, Carey & Carey [4], Cognition and Technology Group at Vanderbilt [5]).

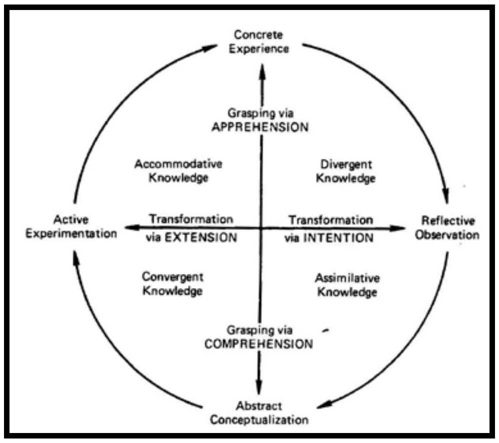


Figure 1. Kolb's Structural Dimensions Underlying the Process of Experiential Learning and the Resulting Knowledge [2, p. 42].

What is posited herein is that by rigorously adhering to Kolb's model for experiential learning in the instructional design process in software intensive courses, the trap of "teaching to the tool" can be avoided. Specifically, if a conscientious articulation of what is intended to be taught is positioned as Kolb's abstraction, and the software functionality is positioned as Kolb's concrete experience, focus is diffused through Kolb's model from the button pushing to the other three critical components of the model: Abstract Conceptualization, Active Experimentation and Reflective Observation. The abstraction takes on the character of a question or hypothesis and the software becomes the vehicle for active experimentation, with the results flowing from the experience and the reflection focused on whether the experience validates the abstraction or the hypothesis. Experiential Learning theorists maintain that learning takes place whether the concrete experience or the abstraction comes first, as long as active experimentation and reflection are involved [2]. However, with engineering's roots in science and the scientific method, beginning the learning cycle with a question/abstraction and then following with the active experimentation, the experience and finally the reflection, most students will find this process familiar and easier to implement.

There are a few things about an ELT approach which should be understood. First and foremost is that it can be "messy". ELT has as its lynchpin the idea that the learning is fostered by developing complex structural connections between current learning experiences and past experiences in order to understand abstract concepts [6]. The path is rarely linear, and the goal is not for the learner to discover and memorize some step by step method within the software but rather to learn how to reason in a software environment. Students who have been previously conditioned to linear learning or to seek the "right" way to accomplish a task will need support in embracing an experiential learning approach. They will need to understand that the process is recursive (iterative) and that finding pathways in the software that do not lead to the solution are as valuable to their learning as those that lead to a solution [7]. By experiencing routines in software that do not produce the specific desired result much can be assimilated into ones experience base for future access when different problems are encountered. By building a rich interconnected understanding of both the abstractions and the "successful" and "unsuccessful" experiences, the results of one's learning is much more versatile than if they had regarded all the experiences not on the linear solution path of the problem as wasteful.

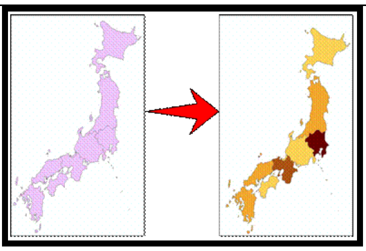
One of the more difficult parts for students working in an ELT environment is in recognizing that what they have always previously characterized as “frustration” with software is actually the learning. Sometimes, for today’s young people, an analogy with computer gaming can help. Asking them: “Why do you keep playing a game where each level gets harder and more complex and you must constantly go backwards and do things over in order to go forward?” can often help them to understand how to manage their frustration. In gaming they see the frustration of a tough scenario or adversary as a challenge. They don’t let it get the best of them, nor do they give up, so if they approach their software intensive learning with the same attitude, they can lower their frustration level immensely. Of course, one substantial difference between gaming and computer intensive learning can be instructor imposed time constraints (deadlines!). But if reasonable exercises are chosen, time management strategies are communicated, and ample access to the software is provided, much can be done to alleviate student frustration that is time constraint based.

Instructors also have to resist the temptation to resort to step by step instruction when working one on one with students. Help often takes the form of Socratic questioning about the abstractions. Focus needs to be placed on helping students understand what they are trying to do rather than walking them through the specific steps. Most software interfaces are constructed to be intuitive once the user knows what it is they are trying to do. If instruction is focused on helping students understand the abstraction, finding the software’s pathway to accomplish it remains secondary. Success requires understanding the abstractions and then answering questions in a dialog box or through command line type entry to elicit a specific behavior from the software. One measure students have of their success is when they can get beyond the “just accept the defaults and hit OK” approach, and are able to look at a complex dialog box and understand the question embedded in each query enough to use their reasoning skills to affirmatively select a response.

The following example demonstrates how an ELT approach to the instructional design differs from a more linear button pushing approach. Figure 2 is an example of a step by step GIS tutorial. It provides a single linear pathway to a specific result. As described above, these are helpful when the underlying concepts are understood but do not provide for adequate connection to abstractions for novice students. Figure 2 is from a free on-line tutorial on how to create a choropleth map in ArcGIS.

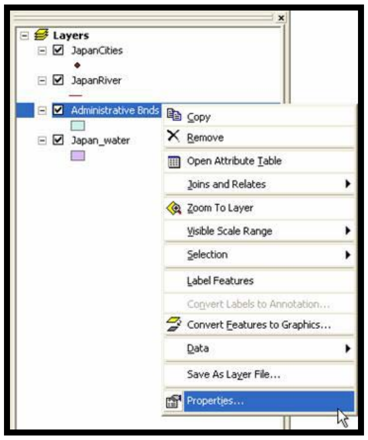
If someone simply wanted to push the fewest buttons to create a choropleth map the tutorial in Figure 2 may very well be appropriate. For instance, an accomplished cartographer or someone who had done similar tasks in another GIS package, might find a single, linear example sufficient to connect their previous experiences and their understanding of the abstractions involved with the new software experience. With such previous experiences and an understanding of the abstractions, a brief tutorial such as the one in Figure 2 can be very effective in getting a new user of the software up and running.

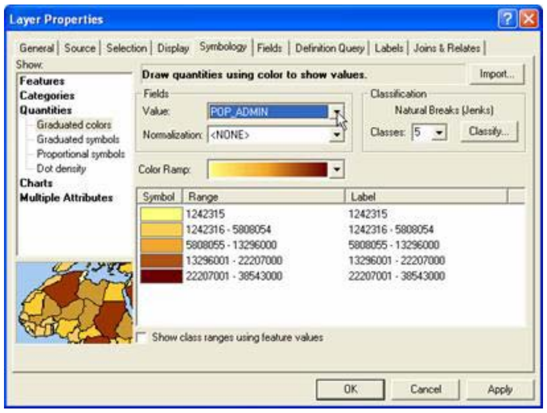
Suppose you have created a layer of *Japan Administrative Boundary* Map from last section. That layer contains data of population of the region and you want to make Choropleth map.



This can be easily done in ArcGIS using the following steps:

1. In the left panel, that is table of content of the layers; do right click on layer *Administrative Boundary* . A pop up menu will be shown.
2. Click on the pop up menu: **Properties**
3. In the Layers Properties dialog, choose **Symbology** Tab
4. Select **Show> Quantities > Graduate Colors**
5. Click field Value: *Pop_Admin*
6. Select the color ramp you want to specify. ArcGIS automatically put the color and category
7. Optionally, you may change the color of each category or change the number of category.
8. Click OK button





Draw quantities using color to show values.

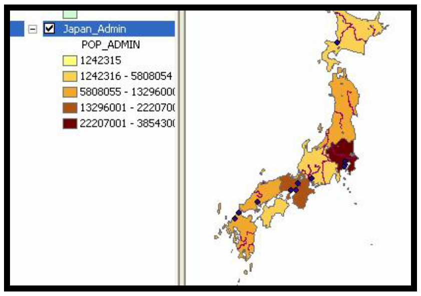
Fields
Value: POP_ADMIN
Normalization: <NONE>

Classification
Natural Breaks (Jenks)
Classes: 5

Color Ramp:

Symbol	Range	Label
🟨	1242315	1242315
🟧	1242316 - 5808054	1242316 - 5808054
🟧	5808055 - 13296000	5808055 - 13296000
🟫	13296001 - 22207000	13296001 - 22207000
🟥	22207001 - 38543000	22207001 - 38543000

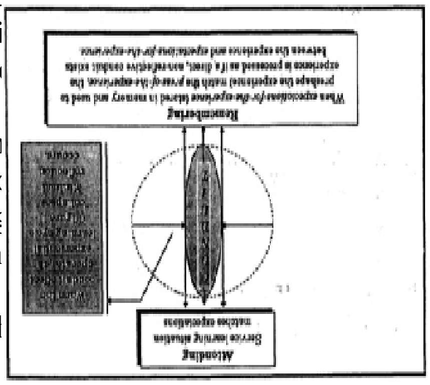
Show class ranges using feature values

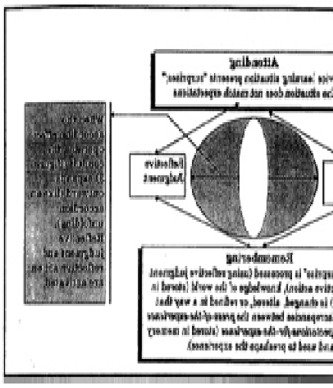


(http://people.revoledu.com/kardi/tutorial/GIS/Choropleth%20map.htm)

Figure 2. Tutorial demonstrating the creation of a choropleth map.

Conversely, without those previous experience based understandings, connections between the button pushing and the abstracti … g to be optimized. Sheckley and Keeton characterized these tw … ve have heretofore seen embodied in training and education) as … fects, respectively [8]. These two modes of learning stem from … e experience based learning process when things are familiar (c … w experiences or abstractions do not fit existing understandin … ences are closely related to previous experiences and the conn … make, the learning is expedited. This is what one might expect … tware package when the abstractions are understood and ot … mastered.





(a) The Conduit Effect (b) The Accordion Effect

Figure 3. Sheckley and Keeton's conceptualization of the conduit and accordion effects as extensions of Kolb's experiential learning model [8, p. 41 and p. 44].

In the second situation, which is much more akin to what we see in undergraduate introductory CAD and GIS education, students must vacillate between the experimentation and the reflection, like an accordion in order to comprehend and understand the relationship between the experiences and the abstractions. This type of learning is generally more time consuming, but this additional time on task allows for the forging of much deeper connections between the button pushing experiences and the results. Figure 3 depicts Sheckley and Keeton's conceptualization of the conduit and accordion effects, and the derivation from Kolb's Experiential Learning concept model from Figure 1 can be noted.

If the training type tutorial on the choropleth map in Figure 2 were to be transformed into one for learning <u>about</u> choropleth maps and how to create them it might include many additional features. For example, rather than the mere series of steps in Figure 2 based on the Layer Properties dialog box an exploration of the other tabs in the Layer Properties dialog box could be included. Even if it wouldn't make sense to explore the complete functionality of each and every tab, at least a definition of each tab would make a connection that is wholly missing in Figure 2. If other tabs had already been explored and such an overview had already been provided, than a reminder of that previous exploration as a segue into the specifics of the Symbology Tab would help in making the connections between the past experience, the abstractions and the current learning experience involving the choropleth map.

In addition, within the Symbology Tab "Show>Quantities>Graduate Colors" involves a number of smaller yet important decisions which should be explored. Restructuring the tutorial presentation to a comparative analysis of the various results from the different choices available on the Symbology Tab, and then ultimately choosing the "Quantities>Graduate Colors" option to create the choropleth map would be one way to do this. The restructuring the tutorial to utilize the ELT learning cycle in the instructional design shifts focus to learning about a matrix of software behaviors and results, rather than a single linear process.

What is being proposed here is a conscious effort in the instructional design process to connect the abstractions to the outcomes using the button pushing as an experience based tool. This stands in contrast to an instructional design method that is focused on defining the most efficient linear process to accomplish a narrow and specific technical outcome. The instructional design approach suggested herein is content neutral and emphasizes seeing and creating relationships between the abstractions, past experiences and the specific learning experiences. <u>It is all about the method not the content</u>. According to Reber the more complex the structure of the relationships that are able to be developed, the easier it is to access the requisite information to address a current problem implicitly [9]. Clark and Elen describe this implicit knowledge as "procedural" knowledge and indicate that between 70% and 90% of the knowledge which is brought to bear on solving a complex problem is this type of knowledge; the stuff we "just do" or "just know" [10]. With this being the case, the importance of developing learning experiences that have as a primary objective converting new knowledge to this implicit knowledge while doing complex tasks cannot be overstated.

Two additional benefits of this ELT approach to learning are that it mimics practice based learning and provides the learner with essential skills for lifelong learning. Practice based learning is characterized by complex problems, with multiple solution paths which are often completely unknown to the expert practitioner before beginning the problem. Complex practice problems rarely follow the solution paths of a specific previous problem requiring the practioner to draw from numerous past experiences and understandings of abstractions to tease out a solution. Ertmer and Newby's expert learning model, like Sheckley and Keeton's [8] conduit effect acknowledges that the more information that is available and can be easily accessed, the more simplified the reflection process and the more efficient the learning process becomes [11]. By utilizing an experiential learning model in instructional design in software intensive engineering education, the skill of forging connections between the abstractions the past experiences and the problem at hand can be honed. Not only does ELT provide a useful framework for instructional design in software intensive education but an ELT approach to engineering design in general may be a fringe benefit inuring to the lifelong learning needs of our students.

References

[1] Tight, M. (2002). Key Concepts in Adult Training and Education 2nd. New York: Routledge.

[2] Kolb, D. A. (1984). *Experiential Learning; Experience as the Source of Learning and Development*. Upper Saddle River, NJ, Prentice Hall, Inc.

[3] Zull, J. E. (2006). Key Aspects of How the Brain Learns. In S. Johnson & K. Taylor (Eds.), *The Neuroscience of Adult Learning* (Vol. 110, pp. 3-10). San Francisco: Jossey-Bass.

[4] Dick, W.O., Carey, L. & Carey, J.O. (2004). *The Systematic Design of Instruction, 6th Ed.* Boston: Allyn & Bacon.

[5] Cognition and Technology Group at Vanderbilt (1990). Anchored instruction and its relationship to situated cognition. *Educational Researcher, 19 (5), 2-10*.

[6] Gentner, D., & Holyoak, K. J. (1997). Reasoning and learning by analogy: Introduction. *American Psychologist, 52,* 32.

[7] Sheckley, B. G., Allen, G. J., & Keeton, M. T. (1993). Adult learning as recursive process. *Journal of Cooperative Education, XXVIII*(2), 56-67.

[8] Sheckley, B. G., & Keeton, M. T. (1997). Service learning: A theoretical model. In J. Schine (Ed.), *Service learning: Ninety-sixth yearbook of the National Society for the Study of Education* (pp. 32-55). Chicago: The University of Chicago Press.

[9] Reber, A. S. (1989). Implicit learning and tacit knowledge. *Journal of Experimental Psychology: General*, *118*(3), 219-235.

[10] Clark, R. E., & Elen, J. (2006). When less is more: Research and theory insights about instruction for complex learning. In J. Elen & R. E. Clark (Eds.), *Handling Complexity in Learning Environments: Theory and Research.* New York: Earli.

[11]Ertmer, P. G. and T. J. Newby (1996). The expert learner: Strategic, self-regulated, and reflective. *Instructional Science 2*: 1-24.

The author is an Assistant Professor of Civil Technology/Surveying and Mapping at the Thompson School of Applied Science at the University of New Hampshire, Durham, NH and is also a doctoral candidate in the Department of Natural Resources at the University of Connecticut, Storrs, CT. This paper discusses the theoretical underpinnings of the instructional design segment of the author's doctoral dissertation research. He can be reached at Cole Hall, 291 Mast Road, University of New Hampshire, Durham, NH, 03824-4728; (603) 862-1687; a.r.vannozzi@unh.edu .