2006-2312: APPLICATION OF OBJECT SCAFFOLDING TO DEVELOP A HANDS-ON, PROBLEM-CENTERED, AND PROJECT-BASED FRESHMAN MATLAB® COURSE

Robin Hensel, West Virginia University

Ye Sun, West Virginia University

Application of Object Scaffolding to Develop A Hands-On, Problem-Centered, and Project-Based Freshman MATLAB® Course

Abstract

An application of object scaffold pedagogy to the teaching of MATLAB® to freshman engineering students is being implemented as part of a problem-centered course during the spring 2006 semester. Object scaffolding has been proposed as a pedagogical technique in which student learning is anchored by a conceptual map resultant from previous learning and in which students are given necessary new information at their point of need.[1] The primary tenants of the scaffolding learning theory were used in the development of a hands-on, problem-centered and project-based freshman MATLAB® course.

The newly developed course is the second in a two-course sequence designed for all freshmen engineering students as part of the common freshman engineering experience. Previously, the two course sequence consisted of a "problem-solving and design" course, followed by a "programming" course. The two-course sequence has been redesigned to carry the unifying concept of the problem-solving and design process throughout both semesters. MATLAB® is taught by presenting students with problems that would require the use of a mathematical programming tool to reach a solution. In this scenario, students, working in teams, are motivated to learn the syntax and structure of the language by the need to solve problems, and therefore, view the software as a problem-solving tool. Evidence of content mastery is assessed, primarily, through evaluation of the quality of student projects.

The application of the scaffolding pedagogy in the "new" course is evidenced by: (1) presenting programming concepts and MATLAB® within a technical problem solving context; (2) fostering increased cognitive development through collaborative interaction among students; (3) building on familiar concepts by beginning with scalar operations, then moving to vector and array operations; and (4) using a text designed to support this pedagogical method.

Introduction

Engineering majors at West Virginia University (WVU), take three engineering courses as part of a common freshman year experience. A one-credit hour engineering orientation course and a two-credit hour engineering problem-solving course are taken in the first semester and a second, three-credit hour, engineering problem solving course is taken during the second semester. The focus of this paper is the re-design of the second semester engineering course from a traditional programming course to a hands-on, problem-centered and project-based technical problem-solving course which uses software as a problem-solving tool. The new course is the result of identifying desired learning outcomes, examining the WVU freshman program, as well as the first year engineering programs of other institutions, and reviewing educational literature, ABET guidelines, and "best practice" articles.

Background

Prior to Academic Year (AY) 2004-2005, several engineering departments within West Virginia University indicated their dissatisfaction with the current second semester engineering course and questioned its relevance to, usefulness for, and necessity in the various engineering majors offered at WVU. During AY 2004-2005, each department was surveyed to determine what knowledge, skills and abilities are needed by sophomore students who enter their engineering discipline major. The items on the resulting list were discussed, refined, prioritized, and, finally, categorized as basic skills, fundamental engineering topics, and computer tools. The first year experience was then reviewed in relation to those elements and several changes were recommended and made to the freshman engineering program.

No significant changes were made to the one-credit engineering orientation course, which provides an excellent introduction to college life and engineering disciplines and careers, as well as teaches study skills and time management practices. The other two freshman engineering courses were modified, as needed, to form a two-course sequence with a problem-solving emphasis. Both courses use individual assignments and team projects to teach fundamental engineering topics, basic professional skills, and the use of a variety of computer tools. Because these elements were, essentially, the focus of the first course, few changes were made to that course. Significant changes were made to the second course. The resulting second course, ENGR 102, is described in this paper. The revised ENGR 102 course was approved by the WVU Faculty Senate during the spring 2005 semester and is being implemented for the first time during the Spring 2006 semester.

Learning Theory

The pedagogical approach for teaching ENGR 102 is an implementation of the Object Scaffolding pedagogical technique in which "student learning is anchored by a conceptual map resultant from previous learning."[2] Object Scaffolding is rooted in the Social Constructivism learning theory.

Constructivism is based on a subset of research within cognitive psychology with a basic premise that an individual learner must actively "build" knowledge and skills.[3] While all constructivists agree that it is the individual's processing of stimuli from the environment and the resulting development of cognitive structures that produce learning, social constructivists believe that those cognitive structures are developed through social interaction.[3] Vygotsky's social development theory of learning proposes that students learn by interaction with more capable members of the same culture, usually teachers or other students, and advocates collaboration among teachers and students.[1,4]

The American Association for the Advancement of Science's Project 2061,[5] challenges traditional instruction by criticizing its failure "to encourage students to work together, to share ideas and information freely with each other, or to use modern instruments to extend their intellectual capabilities."[6] Under a constructivist perspective, the role of the teacher is redefined. Free from the role of information-giver, teachers become facilitators or coaches who assist students to construct their own conceptualizations and develop new insights and connections with previous knowledge.[6,7] They organize information as conceptual clusters of problems, questions and discrepant situations in order to engage students' interests on a topic that has a broad concept.[8,6] In addition, Constructivism promotes curriculum that is customized to students' prior knowledge and interest, and emphasizes problem-solving.

Constructivists recommend that learners be provided with the opportunity for learning with an increasing complexity of tasks, skills and knowledge acquisition. Collaborative and cooperative learning environments are favored in order to expose the learner to alternative viewpoints. Real-world environments that employ the context in which learning is relevant should also be created. Researchers describe several goals for the constructivism learning environments, such as providing experience in and appreciation for multiple perspectives; embedding learning in realistic and relevant contexts; and encouraging the use of multiple modes of representation within the knowledge construction process.[9,10,11] These goals can be achieved by assigning students to work collaboratively, in small groups, on a series of well-designed problems.

When students work collaboratively to solve problems, they learn from each other, but occasionally encounter difficulty and frustration at a specific point. They are, at that point, on the verge of understanding, but need assistance from a more capable member of the learning community. That point of need is called "the state of proximal development"[12] and the term "scaffolding" refers to the act of providing assistance to a learner at his or her point of frustration with a problem and need for information.[1] In industrial engineering terms, scaffolding is "just in time" information delivery. The student is motivated to learn what the instructor is willing and able to teach. By presenting students with problems of increasing complexity, the students move from familiar knowledge to acquiring new knowledge and skills by a set of scaffolding events. In the Vygotsky sense, the goal of scaffolding is to get the learner to attain mastery in solving a variety of target problems and to demonstrate that mastery by solving problems without help.[1]

While constructivist approaches are somewhat new to the university teaching environment, they have been used as the theoretical foundations for K-12 math and science educational practices related to teaching, curriculum, environment and assessment. If students have been exposed to a constructivist learning environment, they take responsibility during the learning process[6,13] and should be able to apply their knowledge to real life situations, which, of course, is the goal of the introductory engineering classroom.

Course Description

Based on an analysis of the outcomes of the departmental survey and discussions with the faculty of the various engineering departments within WVU, three basic types of knowledge, skills and abilities were identified as essential preparation for success in the sophomore year. Entering sophomores are expected to have some knowledge of fundamental engineering topics in addition to the foundational math, chemistry and physics knowledge taught elsewhere in the university. This knowledge includes the engineering approach to problem-solving, the design process, basic programming concepts, principles of engineering drawing, ethics, estimation, and basic mathematical, probability and statistical knowledge needed to do elementary data analysis. Basic skills to be developed during the freshman year include: teamwork, organization and project management, technical report writing, presentation skills, essential programming skills, and the ability to understand and apply ethics to personal and professional decisions. Since computers are essential tools to the engineering professions, students also need to know how to use the computer as a tool to assist in oral and written communication, data analysis, computation, and design.

In the recent past, ENGR 102 was taught much like a traditional computer programming course with limited emphasis on engineering problem-solving. Two languages were introduced (C++® and MATLAB®), but students did not learn how to use either of the languages well. In addition, the departmental survey indicated that students need additional practice in solving engineering problems, working effectively in teams, writing technical reports, and making presentations before they enter their chosen majors.

Using the list of desired learning outcomes for the freshman year experience, the revised ENGR 102 was designed to prepare students for an engineering career by providing opportunities to apply mathematics to solve engineering problems, acquire and refine team working skills, practice written and verbal communication skills, enhance problem solving and design skills, and use a computer as a tool for analysis, modeling, design, and communication. MATLAB® was selected as the programming, modeling and analysis tool used in this course. The revised ENGR 102 course continues to emphasize the engineering problem-solving, technical writing, and teamwork principles introduced during the first semester, but focuses on the solution of engineering problems which require students to write programs in MATLAB® as part of the overall problem solution. The engineering problems provide the context and motivation for learning a computer tool that is widely used in engineering. Together, ENGR 101 and ENGR 102 form a broad-based freshman year preparation for entrance to any one of the College's majors.

Course Pedagogy

Clearly traditional methods of instruction would be an inefficient way to attempt to meet the demands of such a diverse set of goals for an introductory course. While traditional lecture and test pedagogy facilitates learning at the lower end of Bloom's taxonomy of cognition (knowledge, comprehension and application), many of the educational activities described are in the higher domains of Bloom's taxonomy: analysis, synthesis and evaluation.[14] The students are expected to analyze a technical problem, define an approach to solve the problem, use appropriate technology and other tools to design or compute a solution, evaluate the result to determine if it is appropriate, correct, efficient and feasible, and finally summarize their work, efficiently and effectively, to an audience. In addition to demonstrating performance in the higher levels cognitively, the students must demonstrate the ability to perform in higher levels of the Bloom's taxonomy affective domain. Students must cooperate effectively in groups, which includes respecting and valuing individual and cultural differences, as well as accepting responsibility for one's own behavior within the group. Furthermore, students must employ an objective and systematic approach in solving problems, both technical and social, and display a professional commitment to ethical practices in order to complete required projects successfully. To achieve these lofty outcomes, a modified constructivist approach, specifically applying an object scaffolding approach is used.

Throughout the course, students, working in teams, use problem-solving techniques and apply their knowledge of basic mathematics (algebra, geometry, trigonometry, and calculus), science (chemistry and physics) and engineering principles to solve engineering problems. These problems are designed to require the use of a computer programming tool as part of the solution. Shorter problems are used as in-class labs, and longer problems are assigned as multiple-day class and homework assignments, and projects. To develop appropriate computer skills, students practice breaking problems into simpler steps, developing mathematical models and algorithms for solution, representing an algorithm as a flow chart or pseudo-code, and finally, converting the pseudo-code into an appropriate program using Excel® or MATLAB®. Since students have significant experience in using Excel from the first semester course, the second course uses Excel® as an introduction to several early concepts to facilitate a constructivist and scaffolding approach of using the student's current conceptual map as a starting point on which to build mastery of new knowledge. Finally, students document their work through technical reports, posters, and oral presentations.

The application of the constructivist learning theory and scaffolding pedagogy in the course is evidenced by: (1) presenting programming concepts and MATLAB® within a technical problem solving context; (2) fostering increased cognitive development through collaborative interaction among students; (3) building on familiar concepts by beginning with scalar operations, then moving to vector and array operations; and (4) using a text designed to support this pedagogical method. The priorities of teamwork, technical writing, presentation, and the problem-solving process are further evidenced in ways student work is assessed.

First, by presenting programming concepts and MATLAB® within a technical problem-solving context, students gain hands-on experience, not only in using a computer language to solve a specific problem, but in grappling with sometimes vague problem-statements and refining them to the specific problem to be solved. Unlike the traditional method of presenting 50 minute lectures on programming structures and syntax, students are given specific technical problems to solve. The solutions require them to apply engineering problem solving techniques to analyze the problem, develop an algorithm, write the code, use the solution provided by the program to answer the original question. While some introductory explanations of programming structures and syntax are necessary, they are presented, at the point of need for most students, in the form of mini-lectures and through the aid of the instructor and course assistants. Larger projects require solutions to be written as technical reports and presented either as a Power Point® presentation or as a technical poster. In addition to the larger projects, several smaller problems, illustrating discrete concepts, are assigned as in-class labs or homework to develop student confidence in problem analysis, algorithm development and programming, so each student can contribute effectively within his or her group.

While some students are, at first, uncomfortable with "jumping" into a problem or writing a program without a lot of introductory information, they gain confidence as they learn from each other and from the instructor and teaching assistants throughout the process. Learning by doing, not only is sound pedagogy, but is what happens, frequently, in the life of a professional engineer. Technology changes constantly. Engineers must continually retrain themselves on specific languages and tools as they use the new technology to help them solve a variety of problems.

Second, fostering increased cognitive development through collaborative interaction among students is critical to creating an appropriate constructivist learning environment. Students work in groups to complete three major projects and work in pairs and small groups to complete in-class labs as well as some homework assignments. As groups work on solving problems, the instructor and course teaching assistants circulate throughout the room to answer questions, as they arise. Occasionally, several groups will begin to ask the same type of question. Since much of the class is at a moment of "proximal development,"[12] the instructor can aid several groups at once by providing "scaffolding" in the form of a mini-lecture. These short lectures are effective, since students are motivated by their frustration and desire to solve the problem to pay attention and gain the needed knowledge.

Third, building on familiar concepts to guide students to learn new concepts is key to a constructivist learning environment. ENGR 102 begins with a review of the technical problem-solving methodology and the application of general problem-solving techniques to algorithm development. The first project assignment uses a challenging problem from ENGR 101, and reviews the problem statement, related math model and solution in EXCEL®, then requires students to solve it using the "new" software tool, MATLAB®. In addition, the first problems are scalar, in nature, since most freshmen are very familiar with scalar computations, but do not have a solid grasp of vector or array operations. Students become comfortable with the MATLAB® environment and syntax, which they can use as a concept anchor for acquiring new knowledge.

Fourth, using a text designed to support the scaffolding pedagogical methodology facilitates the step-by-step approach taken in this class. The textbook adopted for use in this course, *An Introduction To Technical Problem Solving with MATLAB® v.7* by Jon Sticklen and M. Taner Eskil (Great Lakes Press, Inc. 2005), is designed to support an object scaffolding pedagogical method. The text "addresses the need to ameliorate the steep learning curve of MATLAB® as perceived by freshman engineering students"[1] by breaking the task of MATLAB® learning into three independent units: scalar computations, vector computations, presented as a generalization of the familiar scalar operations, and finally array computations, presented as a generalization of array computations. The presentation of the material supports the application of constructivist learning theory as evidenced in both object and process scaffolding pedagogy by leading the student, in small, distinctly defined steps, from the familiar "calculator-like" scalar operations of MATLAB® to the more complex array operations. The text also presents the basic programming concepts within the context of solving a variety of technical problems and uses examples from a wide range of engineering applications.

Fifth, the way student achievement in the course is assessed reflects the collaborative and process-related goals of a constructivist learning environment as well as the overall outcomes of each project. Since a stated goal of the course is collaboration, students are encouraged to assess their own contribution to the group effort, as well as the contributions of others in their group. These self-assessments and group member assessments are used to make adjustments to individual grades. Assessment of team projects and individual assignments include both formative and summative assessments. Teams review and assess the work of other teams, as well as perform self-assessment of their final product and group dynamics. Those assessments are used as part of the instructor's assessment of student work.

The grading structure also reflects the emphasis on collaborative effort, with over half of a student's grade being determined by group projects and assignments, while only one-fourth of the grade reflects individual achievement evidenced by the more traditional tests and quizzes.

Course Assessment

Several indicators can provide a measure of the overall course effectiveness, student mastery of the specified knowledge, skills, and abilities, and student attitudes toward using MATLAB® as a problem-solving tool, and engineering in general. Student evaluations, both quantitative and qualitative, can provide indicators of the course and instructor effectiveness, as well as provide valuable feedback for the continual improvement of course structure, content and material. Cognitive development and content mastery can be measured by the assessment of student work, and a variety of tools exist to measure attitude.

While these measures provide a level of assessment of the course, a true measure of the course effectiveness is the performance of students in key sophomore courses in each of the discipline majors, since the purpose of the courses of the common freshman year is to prepare students to enter, and succeed in, the engineering discipline major of their choice.

Conclusion

Through the use of a survey of the WVU engineering departments, three types of knowledge, skills and abilities were identified as being essential to preparing for a successful sophomore year, as well as important for the rest of a student's academic and professional career. These essentials include knowledge of fundamental engineering topics, such as technical problem-solving and design, basic professional skills in teamwork, organization and project management, technical reporting and presentation, and the abilities to understand and apply ethics to personal and professional decisions and to use a variety of computer software products as engineering problem-solving tools to assist with data analysis, computation, and design, as well as with oral and written communication.

Using the identified needs as learning goals, a second-semester engineering problem-solving course was developed by applying the primary tenets of the Object Scaffolding pedagogical technique and Social Constructivism learning theory. The resulting project-based course teaches basic programming skills using MATLAB® through the solution of a variety of technical problems. Students work collaboratively to solve increasingly complex problems throughout the semester. Early problems involve scalar operations, and move to vector and array operations, after the student is familiar with the basic MATLAB® environment. Students practice professional skills, such as working in teams, project management, technical reporting, and solution presentation as part of the overall problem-solving process.

The new course provides a solid foundation in fundamental skills needed for beginning engineering students to academically succeed and professionally prepare for challenges in a technologically changing world.

Bibliography

1. Sticklen, J., Amey, M., Eskil, T., Hinds, T., & Urban-Lurain, M. (2004, June 20-23). Application of object-centered scaffolding to introductory MATLAB. Paper presented at the American Society of Engineering Education Annual Conference & Exposition, Session Number 3553, Salt Lake City, Utah.
2. Sticklen, J. & M. Taner Eskil. (2005). An Introduction To Technical Problem Solving with MATLAB® v.7. Great Lakes Press, Inc., pp. xvii – xxv.
3. Huitt, W. (2003). Constructivism. Educational Psychology Interactive. Valdosta State University. Retrieved [01-17-2006] from http://chiron.valdosta.edu/whuitt/col/cogsys/construct.html.
4. Riddle, Elizabeth M. EDIT 704, 3/8/99. *Lev Vygotsky's Social Development Theory*. Retrieved [01-17-2006] from http://chd.gse.gmu.edu/immersion/knowledgebase/theorists/constructivism/vygotsky.htm.
5. American Association for the Advancement of Science (AAAS). (1990). Science for all Americans. New York: Oxford University Press, pp. xvii.
6. Hanley, S. (1994). NSF Cooperative Agreement No. DUE 9255745 on constructivism. Retrieved [01-17-2006] from http://www.inform.umd.edu/UMS+State/UMD-rojects/MCTP/Essays/Constructivism.txt
7. Clark, D. (1999). Constructivism. Retrieved [01-17-2006] from http://www.nwlink.com/~donclark/hrd/history/constructivism.html.
8. Keirouz, M. C. (1996). Constructivism in Teaching Mathematics at UPRH, presentation at Joint Mathematics Meetings, January 10 – 13, 1996, Orlando, Florida.
9. Clements, D. H. (1997). Constructing Constructivism. Teaching Children Mathematics, 4, 198-200. 1997. Retrieved [01-17-2006] from http://www.terc.edu/investigations/relevant/html/misconstructing.html.
10. Honebein, P. (1996). Seven goals for the design of Constructivist learning environments. In B. Wilson, Constructivist learning environments, pp. 17-24. New Jersey: Educational Technology Publications.
11. Murphy, E. (1997). Constructivist Learning Theory. Retrieved [01-17-2006] from http://www.stemnet.nf.ca/~elmurphy/emurphy/cle2b.html
12. TIP: Theories. *Social Development Theory (L. Vygotsky)*. Retrieved [01-17-2006] from http://tip.psychology.org/vygotsky.html.
13. Baroody, A. (1987). Children's Mathematical Thinking. New York: Teachers College Press.
14. Clark, D. (1999). Learning Domains or Bloom's Taxonomy. Retrieved [01-17-2006] from http://www.nwlink.com/~donclark/hrd/bloom.html