Active Learning in Two First-Year Engineering Core Courses

Darrin M. Hanna, Richard E. Haskell, and Michael P. Polis
School of Engineering and Computer Science
Oakland University
Rochester, Michigan 48309

ABSTRACT

In the fall of 2005 a new 6-course, 21-credit core curriculum was introduced in the School of Engineering and Computer Science at Oakland University. One feature of the new core curriculum is the use of active learning techniques including group work and problem-based learning. This paper will present the overall core curriculum and describe our experiences in engaging the students in active learning and problem-solving activities in the core course ***Computer Problem-Solving in Engineering and Computer Science*** *and our experience thus far in a second active, problem-based core course* ***Introduction to Electrical and Computer Engineering****.*

Introduction

Retention of engineering students past the freshman year is a nationwide problem [1 – 7]. The attrition rate in engineering at Oakland University is comparable to the national rate, between 40% and 50% [1]. The percentage is even higher for female and underrepresented minority students [2, 7].

A major reason that students leave engineering after the freshman year is that they don't see the relevance of the math and physics courses taken in the first two years to the exciting field of engineering where they can demonstrate their creativity by designing and building new devices, products, and systems [8]. Traditional freshman engineering courses contribute to this exodus by often discouraging innovation and forcing students to adhere to rigid rules. While following standard procedures is necessary in educational as well as corporate and government environments, it is also important to encourage creativity and innovation from which new technologies and economic growth will arise.

After a comprehensive review of our old 28-credit engineering core curriculum, a completely new 21-credit engineering core curriculum was introduced in the School of Engineering and Computer Science (SECS) at Oakland University in the fall of 2005. The purpose of this new core curriculum is to provide all engineering students with an enhanced opportunity to:

- be exposed to a broad view of engineering early in the curriculum,
- acquire the fundamentals of computer, electrical, mechanical, industrial and systems engineering,
- develop their problem solving skills,
- develop their laboratory skills,
- use modern engineering tools early in the curriculum,
- develop their oral and written communication skills,
- participate in the design process through a significant team design experience during the sophomore year.

This new core curriculum contains the following six courses.

EGR 120 Engineering Graphics and CAD (1 credit)
EGR 141 Computer Problem Solving in Engineering and Computer Science (4 credits)
EGR 240 Introduction to Electrical and Computer Engineering (4 credits)
EGR 250 Introduction to Thermal Engineering (4 credits)
EGR 260 Introduction to Industrial and Systems Engineering (4 credits)
EGR 280 Design and Analysis of Electromechanical Systems (4 credits)

The first course is a 1-credit hands-on course in *Engineering Graphics and CAD* that was offered for the first time in the Fall 2005. The purpose of the course is to excite students by having them learn what can be done with modern, industry-level CAD tools. The first 4-credit course, *Computer Problem Solving in Engineering and Computer Science*, is the only one of these courses that has been taught for the past five years using active learning and problem-solving. All other courses are completely new core courses. The first section of this paper describes our experience in teaching this course for the past five years. The third core course, *Introduction to Electrical and Computer Engineering*, was offered for the first time in the Winter 2006 term starting in January 2006. The second section of this paper describes how problem-based learning and other active-learning activities drive this course.

The remaining three core courses will be offered for the first time in the fall of 2006. *Introduction to Thermal Engineering* will combine the most important material from a previous core course on Thermodynamics with new material on heat transfer and applications. *Introduction to Industrial and Systems Engineering* will include substantial material on engineering probability and statistics as well as new material on engineering economics and simulation in manufacturing. The final new core course, *Design and Analysis of Electromechanical Systems*, will serve as a kind of sophomore capstone design course for the core. The first third of the course will cover microprocessor interfacing using C with an emphasis on controlling DC motors and servos. This will be followed by mechanical engineering topics including statics, dynamics, gears and linkages. Students will work in groups to complete a project involving a microprocessor-based electromechanical system.

EGR 141, Computer Problem Solving in Engineering and Computer Science

The first 4-credit core course, *Computer Problem Solving in Engineering and Computer Science*, was introduced in the fall of 2001 which is also taken by all computer science majors and has the following catalog description.

EGR 141 Computer Problem Solving in Engineering and Computer Science (4 credits)
General methods of problem solving and principles of algorithmic design using a high-level language such as Visual Basic .NET. Introduction to MATLAB. Applications will be drawn from problems in mechanical, electrical and computer engineering and computer science. Laboratory.

Students learn to design, implement, and test algorithms using Visual Basic .NET (originally Visual Basic 6.0) and MATLAB to solve problems in various disciplines of engineering and computer science. Visual Basic .NET is used to teach event-driven programming with a graphical user interface, basic constructs of programming, algorithm design, and interfacing with a database.

Understanding a programming language is only valuable if the students also develop the ability to write computer programs to solve problems they will encounter in their field. Often when introductory programming courses present complex topics, and do not emphasize how different types of problems can be solved by creating programs, many engineering students conclude that programming is more complicated than it is relevant to their field. Furthermore, those who learn the programming language often demonstrate a poor ability to solve programming problems [9].

Learning for the first time algorithmic thinking, object-oriented programming, and syntax using a language as complex as C++ or Java is rigorous. For most students, it is difficult to develop basic skills for algorithmic thinking while developing an understanding of an object-oriented language, especially for those not majoring in computer science. Historically, programming using functions is more intuitive than object-oriented design for most.

The problem-solving topics presented in class include Design and Uncertainty, Divide and Conquer Techniques, Estimating and Predicting Unknowns, Simulation, and Noise and Filtering. These topics were selected as general techniques found in approaching a broad range of engineering and computer science problems regardless of discipline. Problems from different engineering disciplines are given to students to solve throughout the course using Visual Basic .NET while they are learning to program. For non-computer science students, a course of this type teaches skills necessary to solve engineering problems in their field using a simple programming language. For computer science students, this course prepares them to take a next course in programming using a language such as C++ or Java where more attention can be paid to a more powerful language and object-oriented design instead of basic procedural constructs.

This course is a 4-credit course with a 2-hour lab and a 1-hour problem-solving session. Taught in this manner, the class can be taught effectively to large numbers of students by having the entire class attend a single large lecture. In addition, students attend a problem-solving session with at most fifty students in each section, and a lab with approximately twenty students per section. Prior to EGR 141, we ran three smaller sections of a traditional programming course taught by three different full time faculty who met regularly to coordinate the lecture materials, labs, and exams. This traditional approach gave students an opportunity to learn how to design, implement, and test a program from a faculty member in a reasonably small-sized classroom *lecture* setting for approximately 4 hours per week. They would then practice programming during a 3-hour lab session with a graduate teaching assistant.

For EGR 141, we have always taught only one section of the course using three faculty: one for the lecture, one to conduct the problem-solving sessions, and one to rotate through labs and offer two weekly extra help sessions which are open labs where students can meet with a professor. In this way, students learn to program or are introduced to a problem-solving topic in a larger *interactive* classroom setting for approximately 4 hours per week. The students also work with a faculty member for 1 hour per week in a reasonably small-sized problem-solving setting to expand upon and reinforce the current problem-solving topic through hands-on activities. Finally, students work in the lab for 2 hours per week with a graduate teaching assistant and, for labs with more than 10 students, an attending faculty member who comes to their lab section every two weeks on rotation. They also can elect to go to one or both of the extra help sessions and work with a professor to reinforce their programming and problem-solving skills. This way, students work with faculty in all aspects of the course and are provided with more opportunities for reinforcing what they have learned in the lecture instead of the traditional faculty lecture and graduate student-assisted lab with no additional resources required on the part of the university compared with offering multiple, smaller sections of courses. Figure 1 shows an example of EGR 141 during a typical term where approximately 150 students are registered.

EGR 141 Staff: Prof. (A): Lecture Prof. (B): Problem-solving Prof. (C): Labs and Extra Help Sessions Teaching Assistants (TA)
<u>Lecture</u>, two 1:47 lectures per week: (A) Attended by all students
<u>Problem-solving sessions</u>, four 1-hour sections offered per week: (B) Students register for one of these four weekly sessions
<u>Lab sessions</u>, 10 2-hour sections offered per week: =10 students (TA), >10 students (TA)&(C) Students register for one of these ten weekly lab sections
<u>Open Lab sessions</u>, two 1.5 hour extra help sessions (C) Any student may attend the extra session as needed

Figure 1: EGR 141 Course Components and Faculty/TA Assignments

Course Objectives

A successful student in this course will be able to:

- Solve problems in Engineering and Computer Science
- Design an algorithm and use Visual Basic .NET to develop a program
- Use events in the design and implementation of graphical user interfaces
- Use forms, buttons, textboxes, radio buttons, and listboxes in Visual Basic .NET
- Develop Visual Basic .NET code for functions, loops, decision structures (if, case)
- Use the Visual Basic .NET debugger to watch variables and program execution
- Use memory and storage properly including variables, arrays, sequential files and a database using Visual Basic .NET
- Use Matlab Toolbox functions to solve problems

The Lecture

All students in this 4-credit course attend a common lecture. There are approximately three and a half hours of lecture each week. During a typical semester at Oakland University, between 120 and 190 students register for the course. A typical lecture accomplishes one or more of the following objectives:

1. Designing an algorithm and programming conventions, Visual Basic .NET syntax, and Visual Basic .NET controls and events
2. Problem solving topics
3. Laboratory exercises

In the lecture students are taught how to develop algorithms, how to program in Visual Basic .NET, and how to use Visual Basic .NET controls and events. The course introduces Visual Basic .NET syntax including conditional statements, loops, functions, one- and multi-dimensional arrays, file input and output, and using databases with Visual Basic. The controls covered in the course include the textbox, command button, label, frame, radio button, check box, combo box, list box, timer, Microsoft Chart, TrackBar, and Microsoft Comm controls. Typical events are presented such as the Lost Focus, Got Focus, Change, Click, and Timer events. As an introduction to another language we also teach three to four lectures of MATLAB. This includes matrices as data structures, basic MATLAB syntax, how to use toolboxes, and creating MATLAB functions. Students use MATLAB to write functions that perform and compare various-degree curve fits to laboratory data and to add noise to and filter noise from images using the image processing toolbox. MATLAB will be used in successive core engineering courses and serves as a good example of a language/tool for students to compare and contrast with Visual Basic .NET.

The programming lectures are taught interactively. New constructs are presented to students followed by interactively creating a small application that incorporates these new items. The instructor presents the program specifications and using a laptop projecting onto a screen has the students in the class guide the development. Many students participate in this interactive programming. Some bring laptops to class and follow along. Others are urged to download the application from the course website after class and try to build it themselves.

EGR 141 is not taught merely as a programming course; instead, both Visual Basic .NET and MATLAB are presented as tools for solving problems. In other words, topics are presented for students to be able to use programming for developing applications that solve problems and not for students to learn every aspect of the language. For example, instead of teaching the many ways one can perform file I/O, we teach students how to read from and write to files using buffered streams. This pedagogy makes it possible to introduce in a single semester how to develop an algorithm, syntax ranging from basic constructs to using databases, and problem solving topics. Students are encouraged to learn more about Visual Basic .NET and MATLAB on their own.

Five or six lectures during the term are devoted to introducing a new problem-solving topic. Typically, an introduction to the topic is given with an example or two including interactive class discussions where students give examples of situations in their lives where they have used the problem-solving skill. The topic is expanded upon in the problem-solving sessions and additional examples are given leading to a homework assignment that students complete on their own. Each laboratory exercise is also introduced in the lecture. The laboratory exercises drive the lecture topics; the problem-solving topic, controls, events, and Visual Basic syntax necessary to complete the next lab successfully is the lecture agenda. Each lecture offers an interactive approach to learning programming.

Problem Solving

Problem solving is the cornerstone of the approach for teaching this course. Since the students are engineering and computer science students, using problems in engineering and computer science as a basis for developing computer programs prepares them for using Visual Basic .NET or other programming languages to solve problems in future courses. Students register for a problem-solving session when they register for the course. Each problem-solving session has up to fifty students. In these sessions, students learn the problem-solving topics and practice the concepts individually or in small groups. Each problem-solving session is one hour long. A second faculty member (different from the lecturer and lab coordinator) facilitates the problem solving sessions. This instructor provides an example or two, and then instructs the students to form small groups to work on the problem-solving exercises. These problems include hands-on activities ranging from rolling regular and irregular dice for verifying computed probabilities to creating their own filter that they will apply to an image. The problem-solving instructor mentors these groups throughout the session. Problem-solving homework is assigned for each topic and students complete the assignments in groups of two or three. As stated in the syllabus, students who complete the problem-solving assignment without participating in a group will receive a twenty-point deduction. Topics presented in a typical semester include:

1. Design and Uncertainty
   Students are introduced to the iterative process of design including identifying design metrics and parameters. This involves working with parameters that are fixed and those which contain a degree of uncertainty. As an introduction to the topic, we focus on uncertain parameters that follow normal, uniform, and Poisson distributions.
2. Divide and Conquer Techniques
   To solve large problems, students learn when and how to properly divide them into smaller problems that can be solved more easily. These techniques are applied to basic problems in circuit analysis, dynamics, and designing programs through modularity.
3. Estimating and Predicting Unknowns
   Students are introduced to using information that is known to estimate or predict unknowns. Least-squares curve fitting is used to consider how polynomial fits of different degrees can be used to predict missing data. Estimation errors and over-fitting are presented along with examples of problems where statistical methods can be used to fill in information gaps.
4. Simulation
   Combining uncertain parameters with estimation and prediction techniques, simulations can be performed to provide possible scenarios. Students are introduced to performing Monte Carlo simulations for making engineering and business decisions.
5. Noise and Filtering
   Students are exposed to how noise or errors can be filtered out to improve the output of a design including signal-to-noise ratio and information. For students to visualize these concepts, 2D filtering in image processing is used including Gaussian and salt-and-pepper noise, and median, mean, edge, and embossing filters.

A problem-solving topic is introduced approximately every two weeks. During the first week a new topic is introduced with examples, homework assignment, and warm-up exercises. The second week includes a question-answer period along with more advanced practice. Each problem-solving session offers an active learning approach to problem solving.

In the Laboratory

Students spend two hours per week in the lab with between fifteen and twenty students per section. A graduate or undergraduate student is the dedicated lab instructor for the lab section. A third faculty member (in addition to the lecturer and problem-solving instructor) also holds the position of laboratory coordinator. This faculty member participates in labs containing more than ten students on a regular basis by rotating through these sections every two weeks. A student in a lab section with more than ten students would experience a lab environment with a dedicated lab instructor and the assistance of a faculty member every two weeks or so. Additionally, this faculty member offers two 1.5-hour open sessions where any student can attend for extra assistance.

From the Fall 2001 through the Fall 2004, students worked in groups of two in the lab. Each group wrote a lab report after each lab, a traditional mode of operation in labs at the School of Engineering and Computer Science at Oakland University. Although this provides a group atmosphere where students can help each other to complete lab assignments, it also leaves room for one of the partners to dominate. While the group may receive a high score on their lab reports, one of the students may not benefit much from their lab experience. Starting in the Winter 2005 term, we modified the labs where students work individually and focus on developing the algorithm without writing a lab report. Depending on the lab, between forty and sixty percent of the lab assignment is scored through an oral interview with the lab instructor. The lab instructor asks questions specific to their solution, discusses it with them, and assigns a score that indicates how well the student understood their solution and the overall quality of the program. This oral grading requires between half a minute to two minutes per student. We have found that this has worked much better than the traditional approach; each student takes individual responsibility in completing the lab or obtaining help and learning from the helper, students focus on developing the program instead of writing lab reports, and the score and comments that they receive on their lab assignment is through personal contact. This is more motivational for individual improvement than the comments and scores that were previously given through the written lab reports.

From the Fall 2001 term through the Winter 2004 term, the laboratory coordinator would rotate through all of the labs regardless of enrollment. Starting in the Fall 2004 term, we changed this policy such that this faculty member would rotate through labs with more than ten students enrolled and offer extra lab help sessions open to all students. The students are more satisfied with this approach. Offering the extra lab help sessions usually attracts a core of students who regularly attend to reinforce their lab skills along with some students who attend intermittently as needed to finish an unfinished lab or carefully review their lab before their oral-grading interview.

Prior to EGR 141, the introductory programming core course was a more traditional course that used C++ but taught little actual object-oriented design. Table 1 compares a more traditional approach to introductory programming courses to our new approach including EGR 141. EGR 141 teaches basic algorithm constructs, event-driven programming, and graphical user interfaces while covering a broad range of practical topics including databases and serial communication.

Table 1: An outline in changes made to the core programming course

Previous Core Course (1994 - 2001)	New Core Course (2001 – present)
Introduction to computer programming using C++. Topics taught: Constructing a program, data types, functions, loops, decision structures, arrays, standard I/O, file I/O, pointers, linked lists, structures, and an introduction to a C++ class including private and public constructs.	**EGR 141** – General methods of problem solving using Visual Basic. Problem solving topics taught: Fundamentals of probability, circuit analysis, calculating trajectories, numerical methods, and relational database design. Programming topics taught: Events, forms, buttons, textboxes, radio buttons, list boxes, functions, loops, decision structures, debugging in a graphical IDE, file I/O, interfacing with a database, and RS-232 communication to an external device. An introduction to MATLAB.

We have found that students are much more positive about their learning experience overall and are more confident to use Visual Basic .NET to create a computer program than prior to EGR 141. We have also found that it is much easier for students to learn event-driven programming early-on and work with console applications in more advanced courses than vice versa. The interactive class programming integrated with problem-solving in engineering engages students in the lecture. The problem-solving assignments conducted in groups gives students the opportunity to benefit from teamwork. The individual lab assignments and one-on-one oral grading motivates students to do their best work while their lab mentors can provide individual feedback for encouragement and improvement and their scores accurately reflect their individual capabilities. As this employs an interactive learning and teaching style, in order for this course to be successful it is critical to have instructors and lab mentors who enthusiastically deliver a broad range of topics and who are prepared to interact with students in groups or individual mentorship.

EGR 240 Introduction to Electrical and Computer Engineering

The second 4-credit core course, *Introduction to Electrical and Computer Engineering*, is an introduction to the fundamentals of electrical and computer engineering and has the following catalog description.

> **EGR 240 Introduction to Electrical and Computer Engineering (4 credits)**
> An introduction to the fundamentals of electrical and computer engineering; DC and AC circuits; digital logic circuits; combinational logic design; sequential circuits; introduction to electronics; operational amplifiers; DC electromechanical machines. Laboratory.

The text used in the course is *Essentials of Electrical and Computer Engineering* by D. V. Kerns, Jr. and J. D. Irwin, Prentice Hall, 2004. Students use MATLAB, PSpice, and Verilog to analyze and design analog and digital circuits. They also use Xilinx ISE to synthesize logic circuits to a CPLD. In this section we will describe the use of group work, problem-based learning, and oral communication in this course.

Problem-based learning is an educational strategy in which complex, real-world problems are used to motivate students to learn the concepts and principles needed to solve the problem. Problem-based learning is often thought to be more appropriate for small upper-level or graduate courses such as the senior design project, which are open-ended and require considerable initiative on the part of the students [10]. One of the questions that we hope to answer is whether the benefits of problem-based learning can be obtained in large, introductory courses at the freshman level. The idea is to apply the problem-based learning paradigm to engineering design at the freshmen level as a means of learning the fundamentals of electrical and computer engineering.

Students attend three 67-minute lectures plus a 3-hour lab each week. Group work is done in both the lecture (in the form of class homework) and the lab. The content of the course is divided into the following three main topics:

- DC analog circuits and op-amps
- Digital circuits
- AC and electromechanical system

These general topic areas correspond roughly to the three general parts of many real-world designs ranging from toasters to automobiles: 1) user inputs and analog sensors, 2) digital processing of signals and information, 3) outputs to actuators such as DC motors or servos. On the first day of class we describe a device that demonstrates all three areas. For example, this term we described a hand grip that contains a strain gauge to measure force. The result of squeezing the grip is to cause a servo to rotate. Groups of students are challenged to use these or similar components to design a practical device that would provide some useful benefit. This is typical of problem-based learning where students are presented with an open-ended problem in which the solution only develops over time as they learn what they need to solve the problem.

Weekly labs are more structured and focus on specific mini-problems that will illustrate the fundamentals of electrical and computer engineering needed to complete their overall product design. For example, the strain gauge will measure force using a Wheatstone bridge. The small voltage generated is amplified using a differential amplifier using an op amp and converted to a digital signal using an A/D converter that the students design and build in the lab. A digital circuit implemented on a complex programmable logic device (CPLD) together with a D/A converter designed in a previous lab implements a successive approximation A/D converter. A second digital circuit designed and implemented in the lab generates a pulse-width modulated (PWM) signal to drive a DC motor or servo through an appropriate motor driver circuit.

Each week in lab the students in each group are guided to identify questions that identify aspects of their overall problem that they don't understand. By finding the answers to these questions during the following week the students learn to focus their learning efforts.

For the problem-based learning labs to be effective it is essential to have qualified group facilitators serve as lab instructors. We have started to use highly-motivated upper-class undergraduate students for this purpose. Past experience at Oakland University indicates that highly qualified and motivated undergraduates, who have recently taken the course, make better laboratory instructors than graduate students who did their undergraduate work elsewhere. It has also been shown that undergraduate group facilitators can be effective in working with students in introductory problem-based learning courses [11].

The weekly lab assignments contain four questions to be answered by the following week. Each student in a lab group of four students is responsible for submitting a written answer to one of the questions but is also responsible for knowing the answers to the other three questions. During the 3-hour lab session one of the lab instructors will ask each student a question related to one of the three questions they did not submit. The result of this oral quiz counts the same as their written submission of their question. This forces all students to understand all four questions and if a very effective tool to achieve one-on-one interaction with all students in a large class. One group each week presents a PowerPoint presentation of their answers to the four questions to their lab section. The other students in the class rate this presentation on an on-line rating form at the conclusion of the presentation. A summary of these ratings including comments from the students are automatically emailed to the presenters. These ratings do not affect their lab grade and are used only for constructive feedback on their presentation.

Conclusion

Experiences so far show that students are able to handle open-ended problems starting in the first year. Interactive learning during lectures through class-driven programming and class exercises allows students to participate in the teaching/learning process before working on individual or group assignments. By requiring group work and individual work, students receive the benefits of both teamwork and independent thinking and their grades more accurately reflect their individual capabilities. These diverse learning techniques have significantly improved the quality of the course, amount and apparent relevance of material delivered per course, the students' perception of their learning experience, and the students' ability to solve problems over that of our previous traditional lecture-lab approach. So far, the problem-based learning exercise that we have incorporated into EGR 240 has proved to be very beneficial. The use of multiple modern computer tools has significantly enriched the exercises. Although these courses are challenging, we expect that this enhanced learning experience where students solve problems relevant to their discipline and the real-world will also improve student retention in engineering and computer science at Oakland University.

References

[1] M. Anderson-Rowland, "Understanding Freshman Engineering Student Retention Through a Survey," presented at ASEE Annual Conference, Milwaukee, WI, 1997.

[2] L. J. Bottomley, S. Rajala, and R. Porter, "Women in Engineering at North Carolina State University: An Effort in Recruitment, Retention and Encouragement," presented at ASEE/IEEE Frontiers in Education Conference, San Juan, Puerto Rico, 1999.

[3] A. Hammoudeh and J. Barrett, "Tackling Engineering Retention: A Firsthand Experience," presented at International Conference on Engineering Education, Manchester, UK, 2002.

[4] M. I. Hoit and M. W. Ohland, "The impact of a Discipline-Based Introduction to Engineering Course on Improving Retention," *Journal of Engineering Education*, vol. 87, pp. 79-86, 1998.

[5] C. Moller-Wong and A. Eide, "An Engineering Student Retention Study," *Journal of Engineering Education*, vol. 86, pp. 7-15, 1997.

[6] Y. A. Owusu, "Systems Model for Improving Standards and Retention in Engineering Education," presented at ASEE Annual Conference, Albuquerque NM, 2001.

[7] C. Morrison, K. Griffin, and P. Marcotullio, "Retention of Minority Students in Engineering," *NACME Research Letter*, vol. 5, pp. 1-20, 1995.

[8] E. Seymour and N.M. Hewitt Talking about Leaving: Why Undergraduates Leave the Sciences Boulder, CO, Westview Press, 1997.

[9] M. McCracken, V. Almstrum, D. Diaz, M. Guzdial, D. Hagan, Y. B-D. Kolikant, C. Laxer, L. Thomas, and I. Utting, "A Multi-national, Multi-institutional Study of Assessment of Programming Skills of First-year CS Students," ACM SIGCSE Bulletin, 2001.

[10] B. J. Duch, S. E. Groh, and D. E. Allen, Eds., *The Power of Problem-Based Learning*, Stylus Publishing, Sterling, VA, 2001.

[11] D. E. Allen and H. B. White, III, "Undergraduate Group Facilitators to Meet the Challenges of Multiple Classroom Groups," Chap. 8 in *The Power of Problem-Based Learning*, B. J. Duch, S. E. Groh, and D. E. Allen, Eds., Stylus Publishing, Sterling, VA, 2001.