AC 2008-1062: INTEGRATION OF COMPUTER BASED PROBLEM SOLVING INTO ENGINEERING CURRICULA

Dianne Raubenheimer, North Carolina State University

Dianne Raubenheimer is Director of Assessment in the College of Engineering and Adjunct Assistant Professor in the Department of Adult and Higher Education in the College of Education at NCSU. She has worked with faculty and administrators in Engineering for two years, and previously in the Science and Education disciplines. She has a background in teacher education, curriculum development and evaluation and has worked as an education consultant for a number of organizations in the USA and South Africa conducting program evaluations. She received a Ph.D. in Educational Leadership and Organizational Development (Higher Education) from the University of Louisville and has M.Ed, M.Sc, B.Sc (Hons) degrees and a postgraduate Diploma in Adult Education from the University of Natal, Durban, South Africa.

Rebecca Brent, Education Designs Inc.

Rebecca Brent is president of Education Designs, Inc., a consulting firm located in Cary, North Carolina. She is also an educational consultant for the College of Engineering at North Carolina State University and co-director of the National Effective Teaching Institute sponsored by the American Society for Engineering Education. Dr. Brent received her B.A. from Millsaps College in Jackson, MS, her M.Ed. from Mississippi State University and her Ed.D. from Auburn University. She was an Associate Professor of Education at East Carolina University before starting her consulting firm in 1996.

Jeff Joines, North Carolina State University

Jeffrey A. Joines is an Associate Professor in the Textile Engineering, Chemistry, and Science Department at NC State University and is the Director of the Textile Engineering Program. He received a B.S. in Electrical Engineering and B.S. in Industrial Engineering in 1990, a M.S in Industrial Engineering in 1990, and Ph.D. in Industrial Engineering in 1996 all from NC State University. He received the 1997 Pritsker Doctoral Dissertation Award from the Institute of Industrial Engineers. He was awarded the 2006 NC State University Outstanding Teaching Award. He also serves as the faculty chair of the student-owned computing (SOC) initiative in the College of Engineering. His research interests include evolutionary optimization, object-oriented simulation, simulation-based scheduling and supply chain optimization. He was the Co-Proceedings Editor for the 2000 Winter Simulation Conference and the Program Chair for the 2005 Winter Simulation Conference and acts as the technical coordinator for the conference's management system. His email and web addresses are <JeffJoines@ncsu.edu> and <http://www.te.ncsu.edu/joines>.

Amy Craig, North Carolina State University

Amy E. Craig is the Coordinator of Student-Owned Computing in the College of Engineering and a doctoral candidate in the Department of Industrial and Systems Engineering at NC State University. She regularly teaches the Introduction to Engineering and Problem Solving course in the First Year Engineering Program. Her research interests include faculty development and teaching and learning in the engineering disciplines. She received her MIE and BSIE degrees from NC State University. Prior to her return to NC State, she worked as a Cost Engineer in the Personal Computing Division of IBM.

Integration of Computer-Based Problem Solving into Engineering Curricula

Abstract

The primary objectives of this engineering project are (1) to examine how to develop students' problem solving and computational skills early in their program of study and (2) to further enhance these skills by building upon critical computing concepts semester after semester. The project is a component of NC State University's quality enhancement plan, which focuses on the use of technology in enhancing student learning. The project stems from new introductory computer-based modeling courses that were created in two engineering departments, and has expanded to include other departments. We give an overview of the project, provide an example of how a problem is modeled and broken apart, present some assessment results, and discuss the emerging lessons being learned.

Introduction

Many engineering curricula around the country are re-evaluating their introductory computer programming requirement. At our university, several departments have changed from the traditional Java or C++ course to something more applicable to their discipline. Realizing that the standard introductory programming course no longer appropriately complements the education of systems engineers, three departments (Textile Engineering (TE), Industrial and Systems Engineering (ISE), and Chemical and Bimolecular Engineering (CBE)) looked at similar approaches to developing or revising existing courses to help students with algorithmic thinking and problem solving using computing.

These courses aim to educate students to model problems relevant to their specific engineering discipline, solve these problems using modeling tools (including a range of software platforms, such as Excel with VBA), and then to analyze the solutions through decision support (i.e., to become "power users" not programmers).

Other departments in the College of Engineering have expressed interest in reviewing their introductory computer programming course requirement and implementing a course similar to those already developed in TE, ISE and CBE. This is the 'scale-out' portion of our project, as we seek to expand the work and develop similar introductory courses in other engineering disciplines. The second part of the project is the 'scale-up' portion, which entails linking computational processes and skills across courses in the curriculum, that is, developing a computational thread at successive levels in program curricula. We acknowledge that not every course lends itself to the use of computational tools, but there are courses at successive curriculum levels where it is appropriate and beneficial to student learning for computational tools to be utilized and problem solving skills to be reinforced. Figure 1 schematically represents these two parts of our project, the 'scale out' and 'scale up' components.

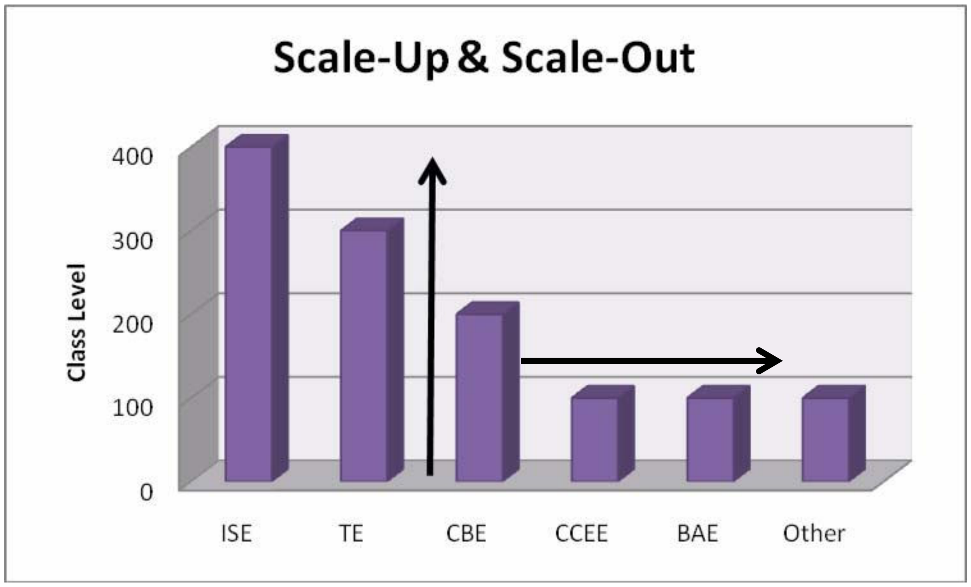


Figure 1: Schematic representation of the implementation process

This project is funded out of the Provost's office and is part of the university Quality Enhancement Plan (QEP) called Learning in a Technology Rich Environment (LITRE),[1] which focuses on the role of technology in enhancing student learning. This project is one of three large technology projects, with the potential for wide-reaching impact across the campus, selected for support in the phase II part of the QEP (July 2007 – July 2009).

This paper provides detail of the introductory courses in three departments (TE, ISE, CBE), including an example from TE showing how a problem is modeled, and then describes how the computational tools are being integrated into upper division courses in these departments. Thereafter, selected assessment results are presented. We conclude by exploring some of the lessons we are learning from the implementation of this project.

Teaching Modeling and Problem Solving

Textile Engineering and Industrial and Systems Engineering: A new course

For TE and ISE, a change was needed from the existing Java programming course because subsequent classes were not using Java and most of the students use Excel and Access to solve problems once they enter the workforce (see Joines, Roberts & Raubenheimer[2] for more reasons). Over the past two years, Jeff Joines and Steve Roberts developed and taught a new Computer-Based Modeling for Engineers course (TE/ISE 110) emphasizing modeling using Excel and VBA. While the design of the course is fundamental to its creation, the teaching and delivery of the course will determine the ultimate success. This is one of the first engineering courses that students take during their college careers, so it is important to engage the students in learning about their discipline. However this engagement must be done in a way that permits multiple instructors and multiple sections to be taught to offer uniformity in computing experiences. One way to engage the student is using in-class assignments and exercises. We refer to these as "in-class labs" to convey the laboratory nature of these exercises.

Essentially, the course (TE/ISE 110) revolves around the labs with very little formal lecture time. Each lab contains three to five problems/case studies that need to be modeled. An occasional 5-10 minute introduction is given to motivate the students to solve the problem, but even the introductions are punctuated with examples that the students should or could implement and run. Lectures are more spontaneous since they arise from *"teaching moments"* which are instances during class when students realize they have a problem and some commentary from the instructor is needed. At those times, students are most open to listening since they have an immediate use for the information. This approach might be called "just-in-time" lecturing. The students are given small tasks that will ultimately lead to solving the entire problem in the case study. For the in-class lab to be effective, it needs to:

1. Occupy the students during the entire two-hour period,
2. Challenge the students to critically think about their responses,
3. Produce questions about the modeling tool or approach,
4. Allow the students to seek help if they have a problem from the teacher, from a teaching assistant, or from another student, and
5. Allow some flexibility for the instructor to use "teaching moments" to elaborate on specific issues.

Students are held accountable by having to answer a series of questions and are required to turn in a subset to be graded. Finally, the students are given homework and projects that are based on real data to test their new-found problem solving skills. Joines, Roberts & Raubenheimer[2] explain in detail the development and implementation of the new computer-based modeling class (TE/ISE 110) including teaching with tablet PCs, etc.

Components of each lab:

- Beginning portion of the lab gives an overview of the problems and topics of the day.
- Students are often instructed to download the spreadsheet for the day from the website, which may have data, code, etc. already available.
- Each part of the in-class lab starts with a background to the problem, followed by a series of steps that have to be performed, with more explanations when needed.
- Intermixed with each of the steps is a series of questions that the students have to answer.
- A subset of those questions are repeated on the front page, which has to be filled out and turned in at the end of the period.

One of the main reasons we switched to this new course was to enhance the students' ability to think critically, develop algorithmic solutions to problems (flow chart out a solution), and develop general problem solving skills. One of the approaches we use to teach engineering problem solving methods is the divide and conquer technique (i.e., breaking up the problem into its smallest elements and solving each of the elements (which is easier) and then reassembling the elements to solve the original problem). The following example illustrates how we teach the divide and conquer technique.

Modeling case study example

The following example is the third of five case studies in the second in-class lab (see Joines, Roberts & Raubenheimer[2] for more information). This particular lab deals with modeling in Excel emphasizing the use of named ranges and implementing engineering equations. The students have just learned about the importance of named ranges and how to create them.

Part 3: Can we predict the value? (Breaking a Problem Apart)

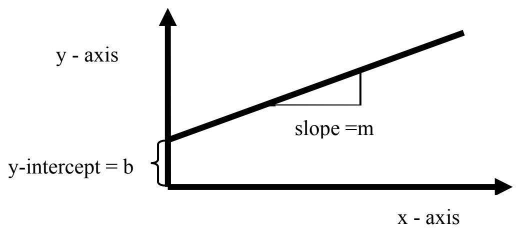
Electronic kits are assembled from various components. The number of labor hours needed to assemble the kit is needed to determine a cost for a kit. However, each kit is different and so the number of hours cannot be directly determined. The company has kept some records on the number of components and the assembly hours required which are given in the worksheet ("Part 3").

Background:

A simple model of the effect of one variable, say *x*, on another, say *y*, is a simple linear equation:

$$y = \mathrm{m}x + \mathrm{b}.$$

Here *y* is the dependent variable and *x* is the independent variable. The parameter "b" is the intercept on the y-axis and parameter "m" is the slope. Graphically, the relationship is:



If that relationship appears appropriate, the question is how to estimate the parameters "m" and "b".

Formulas:

Fortunately, there are statistical methods for this problem called "linear regression" (which you will learn about in statistics classes). The results provide an estimate of *m* and *b* from the following equations:

$$m = \frac{\sum_{i=1}^{n} x_i y_i - n\overline{X}\overline{Y}}{\sum_{i=1}^{n} x_i^2 - n\overline{X}^2}$$

$$b = \overline{Y} - m\overline{X}$$

The x_i and the y_i are the observations of the independent and dependent variables, respectively. The $\bar{X}$ and the $\bar{Y}$ are the averages of the respective variables. Obviously, Excel has functions that we will use later to determine the slope and intercept. As an engineer, you should break a problem into simpler portions and solve each portion separately, which reduces the chance of introducing errors into the final solution (we sometimes call this a divide and conquer strategy). Implementing the above equation all in one cell would not be easy to do correctly.

An Aside: Are you familiar with the Σ sign? It is called the "summation sign" and it means that you want to sum a series. Here are some examples:

$$\sum_{i=1}^{10} i = 1+2+3+\ldots+10 \qquad \sum_{j=1}^{4} x_j = x_1 + x_2 + x_3 + x_4$$

$$\sum_{i=3}^{n} z_i^3 = z_3^3 + z_4^3 + z_5^3 + \ldots + z_n^3 \qquad \sum_{m=0}^{100} 2^m = 2^0 + 2^1 + 2^2 + \ldots + 2^{100}$$

$$\sum_{i=1}^{3}\sum_{j=1}^{2} x_i y_j = x_1y_1 + x_1y_2 + x_2y_1 + x_2y_2 + x_3y_1 + x_3y_2$$

Sometimes the limits of the sum are implied by the applications, such as Σxy, which means to sum the product *x* times *y* over all their values. You'll see examples below.

Step 1: Go to the worksheet named "Part 3". In this worksheet, the computational framework has been created, and it also includes the data on the number of components (x) and the labor hours (y). Copy the formulas for the linear regression from the "Formulas" worksheet.

Step 2: Select all of the *x* and *y* values and name the columns **x** and **y**.

Step 3: Next we will compute *xy* and x^2 for each of the values in columns of **D** and **E** as labeled using the named ranges (**x** and **y**). B

Question 1: In row **2**, what did you get for: x*y: _______ and x^2 ______?

Step 4: Now double click the fill handle to copy formula down the **D2:E2** to row **26**. Name these data ranges using *Insert* → *Name* → *Create*. (Note the name of the x^2 column.)

Step 5: Compute *n* from `COUNT()` and the various sums using `SUM()`. You should use the named ranges to minimize errors (e.g., `sum(x)`).

n: ___ Σy: ___ Σx: ___ Σxy: ___ Σx^2: ___

Step 6: Using *Insert* → *Name* → *Create* to add names in their associated cells for **A29:E29.** Notice the names include the symbols as well.

Step 7: In **C31** and **C32**, compute the average of *x* and *y* using the names just created in the previous two steps. Be sure to add names for these cells.

Step 8: Using the names, create Excel expressions in **C34** and **C35** for "m" and "b" from the formulas given for linear regression.

Question 2: What did you get for m: ______________ and b: ______________?

Step 9: Name these cells as "m" and "b".

Question 3: Write down the complete linear equation that calculates the expected hours (*y*) for a given number of components (*x*):

Question 4: If a kit requires 86 components, how many hours do you expect the assembly to take? _______________

Throughout the lab, students will ask questions that become teaching moments. Also, students are not asked to do all nine steps at once. For this lab, we pace the exercise by asking them to perform a couple of steps which gives the TA, instructor, as well as neighbors, time to help. Then, the students are asked for the answers to get them to interact and keep everyone on pace. The students are amazed that very few errors are introduced since we calculated each individual element (e.g., average of the x values or the sum of the x*y values) and verified these were correct and made sense before trying to implement the entire equation for the slope. The other thing to note is how using named ranges reduces errors in implementing complicated equations like the slope equation. The following table depicts two correct solutions to implementing the slope as an Excel expression.

=(Σ_xy-n*xAvg*yAvg)/(Σ_x2-n*xAvg^2)	Slope equation using named ranges
=(D29-A29*C31*C32)/(E29-A29*(C32^2)	Slope equation without named ranges

The first one utilizes the named ranges that were defined. It is very easy to verify that we implemented it correctly by comparing it to the actual equation from the lab. Contrast this with the second more common way which does not use named ranges. One is not sure exactly what cell **C31** refers too. A person could have easily entered the wrong cell (e.g, **A30** instead of **A29**) or clicked on the wrong cell when entering the expression. Just seeing cell references makes it difficult to verify the correct equation compared to the first one which looks like the actual equation.

Chemical and Biomolecular Engineering (CBE): An existing course

Unlike TE and ISE, CBE majors were no longer required to take an introductory programming course. However, CBE was concerned that the students were not getting practice in algorithmic thinking, something Jan Genzer, Professor of Chemical and Biomolecular Engineering, considered a chief benefit of programming. Like Joines, Roberts & Raubenheimer[2], he is not in favor of programming for its own sake but rather programming as a tool for problem solving. After talking with alumni, he came to the same conclusion that most of them were using Excel as the primary tool for numerical work as opposed to Matlab or process simulation packages like Aspen or Super Pro. He decided to use Excel with VBA in CHE 225 (a numerical methods course) as a way to get students thinking algorithmically and to advance their problem solving skills.

Since CBE did not have a course dedicated to modeling like TE/ISE 110, the programming was integrated into an existing course. Owing to time constraints (i.e., they still have to teach numerical methods), they developed a booklet/CD tutorial called *Visual Basic for Applications (VBA) in Microsoft Excel for Chemical Engineers*.[3] The CD tutorial takes the students through a series of exercises to teach them basic concepts and tools they will need as they use Excel with VBA for problem solving. Several sample spreadsheets can be opened and used, and there are notes and activities to help students learn specific tools. The chapters are:

I. Introduction to Excel

II. Programming and Basic Macros

III. Direct VBA Programming and User Forms

IV. Numerical Methods and VBA

V. Error Checking and Debugging

VI. Applications and Engineering Problems

VII. Notes on Learning Resources

In CHE 225, students have 12 sessions where they bring their laptops to class and use them in structured activities around Excel with VBA. Often the students have downloaded a spreadsheet that may be partially completed to use in the session. This approach is similar to the in-class labs of TE/ISE110, but a little less structured. These classes are where Dr. Genzer stresses algorithmic thinking, flow charts the steps in problem solving, and helps students to think through what needs to be programmed. Students are told that it is not important that they know all the syntax. Instead they are encouraged to utilize the resources in Excel and the web. What is important is that the students understand how to approach problem solving using Excel and VBA as tools. During class, students often collaborate with one another and he circulates to help them as they're working through the programming. He frequently pulls the class together to get them to think about what they are doing, to evaluate whether their answers make sense, and to predict what will happen when they take a certain step. At the end of the semester, students complete a project using Excel with VBA (e.g., build a unit conversion form that will convert all kinds of units from English to Metric, etc.).

As part of the plan to expand this learning, Student Owned Computing (SOC) at the College of Engineering hired several students to revise and update the material for CBE 225. However, the major undertaking was to expand the example uses of Excel with VBA within the curricula for the introductory CBE course (CHE 205), as well as the junior level courses that follow CHE 225. Now, CHE 205 students are introduced to the basics of Excel in their problem sessions led by TA's.

We have been working with the group of faculty who teach CHE 225 regularly to try to reach consensus about what tools will be taught in the class. Previously, Dr. Genzer was the only one using VBA. Other instructors used MatLab and Aspen. Now all instructors are committed to using Excel with VBA and introducing MatLab in the class. Consensus about consistent coverage for all sections has not yet been reached, but faculty have been talking about what they are doing and are moving in the direction of more consistency.

Creating a Computational Thread through Successive Curricula

A second focus of this project is to integrate computational tools for modeling and problem solving beyond the introductory courses, into upper division courses. This component seeks to develop a "computing thread" within each targeted department that promotes incorporation of a set of computational tools and methods throughout the core department curriculum.

As described in a previous section of the paper, the Departments of Chemical and Biomolecular Engineering (CBE), Industrial and Systems Engineering (ISE), and Textile Engineering (TE) have each established a strong foundation of computing in their introductory courses. In CBE students are introduced to spreadsheeting for problem solving (using Excel) in their first course in the major, followed by a course in numerical methods where they learn MatLab and expand their use of Excel to include Visual Basic for Applications (VBA) programming. The emphasis in the second course is on algorithmic thinking and a systematic approach to problem solving. In ISE and TE, students take an introductory course in which spreadsheeting and VBA are used extensively in the context of solving both engineering problems and problems in everyday life such as investment planning and queuing theory.

A deficiency in all three departments is that students receive this intensive introduction to computational tools in their second year and then rarely see the tools again until the senior capstone course. With this much of a time gap between computer applications, students forget what they have learned and are unable to expand their computational skills to address the more complex problems they encounter in the advanced courses. Moreover, the inclusion of computational tools in a course can depend heavily on who is teaching, so that students' exposure to computers can vary considerably from one year to another.

In this project, we are working on two levels to change this situation. First, we want to expand the use of computational tools into most upper level courses on a course-by-course basis. In addition, we want to identify a sequence of courses in each department that includes extensive use of the tools, a "computing thread" that promotes student computing skill development through the curriculum. These two levels go hand-in-hand. Faculty can initially identify courses that incorporate computer tools and the interactions and repetitive use of specific tools in subsequent years will lead to identification of the thread and strengthen its coherence.

Chemical and Biomolecular Engineering

Each department presents unique obstacles to implementation of this portion of the project. In CBE, there is a special challenge in that faculty teach courses in rotation, with as many as six different faculty members teaching a particular course over a three-year period. It is difficult to get agreement among those faculty members on the specific computer tools to be incorporated into the course and for them to gain the required skills to teach the tools when they teach the course so infrequently.

The courses listed in Table 1 have been identified as the CBE Computing Thread. They either now include a significant computational skill element or they are slated to have one.

Table 1. CBE Computing Thread

FALL	**SPRING**
Sophomore Courses	**Sophomore Courses**
CHE 205: Chemical Process Principles	CHE 225: Introduction to Chemical Engineering Analysis
Junior Courses	**Junior Courses**
CHE 311: Transport Processes I	CHE 312: Transport Processes II
CHE 315: Chemical Process Thermodynamics	CHE 316: Thermodynamics of Chemical and Phase Equilibria
	CHE 330: Chemical Engineering Lab I
Senior Courses	**Senior Courses**
CHE 450: Chemical Engineering Design I	CHE 451: Chemical Engineering Design II
CHE 446: Design & Analysis of Chemical Reactors	CHE 331: Chemical Engineering Lab II

In their junior year, students take a two-semester transport processes sequence and a two-semester thermodynamics sequence. These four courses had almost no computer applications incorporated, and so they became the focus of most of our effort in CBE. After meeting in groups and individually with faculty teaching those courses, we decided to set up an online faculty repository of relevant computer problems and their solutions. The website is password-protected and available only to the faculty. Our rationale was that faculty unfamiliar with a particular computer tool are unlikely to have the inclination or the time to develop the new examples and problems that they will need to use it in their courses, while if they have previously verified problems and solutions to assign they will be much more likely to use the tool. In addition, the website should facilitate communication and sharing of problems among all faculty teaching a specific course, which will expand the repository. For example, someone teaching thermodynamics one semester may develop spreadsheet problems and examples and upload them to the website to be used by their colleagues in subsequent semesters. When that faculty member comes back to teach the course several years later, he/she will find an expanded collection of problems to use. Finally, this approach should help to provide continuity from semester to semester.

For each course the following items were collected:

- Links to relevant online resources such as simulations and java applets that illustrate key course concepts

- Computer problems and solutions developed by a student team in collaboration with faculty. For each problem statement and solution, there are supporting materials including learning objectives and teaching suggestions.

- Other problems, including sets of spreadsheets developed by Brice Carnahan, Mark Burns, and Philip Savage of the University of Michigan and distributed at the 2007 CHE Summer School sponsored by the Chemical Engineering Division of ASEE. Most of these spreadsheets are set up as tools to solve a class of problems and are therefore useful in many different situations and can be used repeatedly by students in solving different problems.

In the fall 2007 semester we met regularly with faculty teaching the courses in the Computing Thread to offer assistance, keep them informed about additions to the resource repository, and solicit ideas for actions we could take to help them incorporate more problems using computational tools. This was preceded by gaining necessary support from the department head and the associate department head. We also met with the entire CBE department faculty at the end of the fall 2007 semester to overview the project and encourage full departmental support of the curriculum changes. Future plans are to continue adding problems to the repository and to develop an Excel-VBA tutorial and problem sample set to help students gain the computing skills they will need to solve the relatively complex problems they will encounter in the transport and thermodynamics sequences.

ISE and TE Departments

In both the ISE and TE departments, a smaller number of faculty teach the same or similar courses each year, providing greater continuity in content from semester to semester. Two well-respected senior faculty members in ISE and TE developed the aforementioned introductory computer-based modeling course jointly and have continued to work together to share ideas. They have each served as champions of the project and mentors to their colleagues. Several other ISE faculty teaching related courses have been enthusiastic about working together to ensure students see the connections between particular courses and feel more confident in their computational abilities. The TE and ISE Computing Threads are shown in Table 2 and Table 3.

Table 2. ISE Computing Thread

FALL	SPRING
Sophomore Courses	**Sophomore Courses**
ISE 110: Computer-Based Modeling for Engineers	ISE 216: Manufacturing Engineering Practicum
Junior Courses	**Junior Courses**
ISE 361: Deterministic Models in Industrial Engineering	ISE 401: Stochastic Models in Industrial Engineering
ISE 441: Introduction to Simulation	ISE 443: Quality Control
Senior Courses	**Senior Courses**
ISE 453: Production System Design	ISE 417: Manufacturing Engineering III: Computer Integrated Manufacturing
	ISE 498: Senior Design Project

Table 3. TE Computing Thread

FALL	SPRING
Sophomore Courses	**Sophomore Courses**
TE 110: Computer-Based Modeling for Engineers	TE 201: Textile Engineering Science – Fibers
TE 200: Introduction to Polymer Science and Engineering	TE 205: Analog and Digital Circuits
Junior Courses	**Junior Courses**
TE 301: Engineering Textiles Structures I: Linear Assemblies	TE 302: Textile Manufacturing Processes and Assemblies II
TE 303: Thermodynamics for Textile Engineers	TE 440: Textile Information Systems Design
Senior Courses	**Senior Courses**
TE 401: Textile Engineering Design I	TE 402: Textile Engineering Design II
	TE 404: Textile Engineering Quality Improvement

We have met individually with faculty teaching courses within the ISE and TE Computing Threads intermittently since the start of the project. In ISE, project and departmental funds were used to hire student workers to help faculty develop problems and activities. Some problems developed to date include the use of spreadsheets to create break-even analysis templates, using

VBA in advanced database applications, and converting MatLab code for modeling production systems problems to VBA. We maintained regular contact with the students and faculty during the development process to provide pedagogical and technical support. During the fall semester we met with the entire faculty of TE and ISE to review the project status and garner support and ideas from the faculty as a whole, and have also elicited support from departmental heads. Future plans include establishing more connections between courses in the curriculum to strengthen the Computing Thread and more assessment of the impact of computational thinking on students' problem solving abilities.

Assessment Activities

The project has four main research questions, each being assessed using different instruments.

Overall Research Questions

1. What are student approaches to modeling and problem solving and how do they change over time as students move into upper division courses and use programming and computational tools to model and solve discipline specific problems?

   Instrumentation: (a) Student surveys assessing attitudes and confidence in specific course outcomes, (b) survey questions about modeling and problem solving, (c) specially developed, common problem solving tasks, (d) student reflections about how they go about solving tasks, and (e) course-related samples of student work. The survey data is being compared to students completing these same or similar surveys in subsequent semesters, to establish trends in self-confidence as students are exposed to the new course sequences. The problem solving tasks and reflections are being compared to students in upper division courses who have not been through the new course sequences, as well as to do longitudinal tracking of problem solving abilities of students who do experience the new course sequences.

2. What characteristics (e.g. gender, GPA) do the learners bring to problem solving processes that assist or hinder their success as modelers and problem solvers?

   Instrumentation: (a) Student profile data (GPA, gender, etc.), and (b) student surveys, e.g. the Revised Study Process Questionnaire, and a self-efficacy & beliefs about the course survey. These data will be related to data on student problem solving, student performance in courses and other student surveys.

3. Does student performance in the discipline improve with the new approaches to teaching modeling and problem solving?

   Instrumentation: (a) Course specific end of semester GPA comparisons to the same courses in previous years.

4. How do the various faculty involved in the project use technology inside and outside of the class to enhance student learning?

<u>Instrumentation:</u> (a) Baseline faculty survey, (b) field notes of research participants, and (c) faculty interviews.

Some assessment results

Because of the large scope of the assessment activities, only selected assessment results will be shared in this paper, including some of the survey results and findings from the problem solving tasks.

Surveys

We have used surveys to assess student's attitudes towards the new courses and their confidence levels in using various computational tools. In the TE and ISE 110 courses, data has been gathered each semester since fall 2006. This data is used for faculty to monitor the student's skill levels in successive semesters, and to make changes where there are perceived areas of weakness. For instance, Table 4 shows student ratings of their confidence in using various VBA functions in fall 2006 and again in fall 2007. It was noted in fall 2006 that there were some areas where students had not adequately grasped the skills (such as creating loops, writing event handlers, and developing decision support systems). Thus, more attention was given to these areas in subsequent course offerings, resulting in increased confidence levels among the 2007 group.

Table 4. Results of student confidence survey from fall 2006 and 2007

	TE 110 Fall 2006 N = 24	**TE 110 Fall 2007 N = 20**
Rate your confidence with VBA (1 = not confident, 2 = somewhat confident, 3 = confident, 4 = very confident)	**Mean**	**Mean**
14. Recording macros	3.83	3.95
15. Using Excel objects, methods and properties	3.33	3.65
16. Writing functions and subroutines	3.13	3.37
17. Defining variables of various types	3.25	3.50
18. Making assignments	3.13	3.25
19. Creating loops	2.83	3.25
20. Using 'ifs' and 'cases'	3.17	3.45
21. Creating your own forms and controls	3.50	3.55
22. Writing event handlers	2.71	3.15
23. Developing decision support systems	2.75	2.90

In the department of Chemical and Biomolecular Engineering, baseline data was gathered as students entered the CHE 225 course in spring 2007 and again at the end of the spring semester, just prior to final exams.

Table 5 shows that for CHE 225,

- There was a significant increase on **8/10** Excel dimensions by the end of the course. (The other two dimensions were already high at the beginning of the course).
- There was a significant increase on **10/10** VBA dimensions by the end of the course.

This shows that students had developed the requisite computational skills by the end of the one semester in CHE 225.

Table 5. Comparison of student ratings at the beginning and end of the semester in CHE 225

1 = not confident, 2 = somewhat confident, 3 = confident, 4 = very confident	**Pre-test Spring 2007**					**Post-test – spring 2007**				
	N=55					N=35				
	Percentage					Percentage				
Rate your confidence with Excel	**1**	**2**	**3**	**4**	**Av**	**1**	**2**	**3**	**4**	**Av**
Moving around the worksheet	2	4	24	71	3.64	0	0	17	83	3.83
Entering values and formula	2	7	24	67	3.56	0	0	29	71	3.71
Applying built-in financial, statistical and math functions	16	33	35	16	2.51	3	6	40	51	3.40
Using goal seek	18	35	38	9	2.38	3	6	37	54	3.43
Using solver	29	35	31	5	2.13	0	3	34	63	3.60
Constructing data tables	2	11	40	47	3.33	0	0	29	71	3.71
Constructing graphs	2	5	40	53	3.44	0	0	29	71	3.71
Using pivot tables	60	18	16	5	1.67	37	34	11	17	2.09
Using lists	33	27	29	11	2.18	9	40	23	29	2.71
Using Named Ranges	44	24	24	9	1.98	6	29	31	34	2.94
	Percentage					Percentage				
Rate your confidence with VBA	**1**	**2**	**3**	**4**	**Av**	**1**	**2**	**3**	**4**	**Av**
Recording macros	85	7	5	2	1.24	0	23	34	43	3.20
Using Excel objects, methods and properties	75	13	7	5	1.44	0	46	31	23	2.77
Writing functions and subroutines	78	11	4	7	1.40	9	41	26	24	2.91
Defining variables of various types	78	9	5	7	1.42	6	24	50	21	2.85
Making assignments	83	6	4	7	1.35	17	26	40	17	2.57
Creating loops	80	5	7	7	1.42	17	31	34	17	2.51
Using 'ifs' and 'cases'	78	7	4	11	1.47	14	23	46	17	2.66
Creating your own forms and controls	85	5	4	5	1.29	23	26	34	17	2.46
Writing event handlers	85	5	7	2	1.25	57	20	14	9	1.74
Developing decision support systems	85	9	4	2	1.22	57	20	14	9	1.74

In spring 2007 the same survey was administered to five sections of 300 level courses. All the students in these sections would have taken CHE 225 during the previous year, most of them in the spring and some in the summer. The results showed that,

- **All** groups were significantly **less** confident on **all** VBA dimensions than students at the end of CHE 225.
- There were also significant differences on **a few** Excel dimensions in comparison to the CHE 225 students.

Similar data was gathered in a 400 level course in fall 2007 and showed that

- Students were significantly less confident on **one** Excel dimension and **all** VBA dimensions, in comparison to students at end of CHE 225.

Across the board, in all courses, data showed that students were not very confident in using Matlab, Maple & Aspen.

The difference in student confidence between the upper level classes and the end of CHE 225 can be attributed mainly to the fact that the skills learned in the sophomore course were not reinforced in the 300 and 400 level courses. Another factor is that different instructors taught the spring and summer versions of CHE 225, each with their own particular emphasis.

At the end of 2007, these results were presented to the CBE faculty, to initiate a discussion about what skills students do need to develop, the need to develop a computational thread through the curriculum, and an attempt to reach some agreement about which tools should be used. In this way, the assessment data is being used to drive decisions about the best approach to curriculum development and the development of the 'computational thread' through subsequent courses.

Problem solving task

A generic, common problem solving task that involved modeling and making a decision about a job offer scenario was developed for implementation in the TE and ISE 110 class, and in selected upper division classes (400 level) for each of those departments. Students were required to complete the task individually and independently outside of class over a one week period, and then to turn in all their working, as well as a solution to the problem. On the day they turned in their work, they also completed an in-class online reflection about their problem solving process. The questions were framed to reflect the different problem solving stages implicit in the developmental model for problem solving developed by Wolcott[4]. A scoring rubric was developed to score the student's work as well as the accompanying reflective responses. We are currently analyzing this work, and will present some of the findings at the conference.

Lessons Learned

While we are still early in the project, we have learned a number of lessons about increasing computer utilization in engineering departments.

- *Change is hard and takes more time than expected.* We should all know this by now but it still comes as a shock that it takes so long to make changes happen. While people who are first adopters will readily jump in and try new things, more seasoned faculty take much longer and are much harder to convince that change is even necessary.

- *The culture of each department is very different and must be taken into account.* The three departments we are working with vary in the leadership styles of their department heads, the readiness of the faculty to embrace the changes, and the computational needs of the students. It is important to the success of the project to be sensitive to the differences and flexible in designing the course changes and assessments.

- *Talking about teaching is not a common activity in most departments.* In our initial meetings it was often obvious that faculty had almost no idea about what their colleagues were doing in their classes. One unexpected value of a project like this one is that it gets people talking to each other and sharing ideas.

- *Assessment activities are critical for helping faculty realize there is a problem, become galvanized to take action, and continue to implement changes.* When faculty are confronted with the deficiencies in student confidence and skills levels, they are more receptive to making changes to address them. By the same token, when there is ongoing assessment of activities, faculty are able to see results and are encouraged to continue making the effort to change.

- *Departmental leadership is needed to get faculty to the table.* Department heads have the leverage to encourage faculty to make good faith efforts to modify their courses. It can help the project considerably if the department head strongly supports it and participates in some of the discussions. Also, most new faculty are willing to try new things but department heads and senior faculty need to recognize these as valuable contributions towards the tenure process.

- *Champions within departments are critical to success*. Given how challenging it is to make curricular changes, having respected senior faculty who believe in and participate in the project makes everything go more smoothly. Such faculty can mentor their colleagues as they try new things and can be positive voices in faculty discussions. Champions are often the first to try out new tools and techniques and have the enthusiasm to keep trying even when problems arise.

- *Support for computer problem and activity development is an important element in getting faculty to incorporate computational tools*. The one thing all faculty have in common is not enough time to do everything they need to do. Locating and developing problems and activities takes focused time, time most faculty are not willing to take. We have found that providing problem development and pedagogical support to faculty facilitates their making changes they would be unlikely to attempt without such support.

Bibliography

1. LITRE (2005). *Learning in a technology rich environment: A quality enhancement plan for North Carolina State University*. North Carolina State. <http://litre.ncsu.edu>

2. Joines, J.A., Roberts, S.R., & Raubenheimer, C.D. (2007). Computer-Based Modeling for Engineers using Excel and VBA. *Proceedings of the 2007 American Society for Engineering Education International Conference, Honolulu, Hi*. Downloaded 1/6/2008 from http://www.asee.org/acPapers/code/getPaper.cfm?paperID=13338&pdf=AC 2007Full3009.pdf

3. Genzer, J., & Carnell, B. (2005). *Visual Basic for Applications (VBA) in Microsoft Excel for Chemical Engineers.* NC State University, Department of Chemical and Biomolecular Engineering.

4. Wolcott, S.K. (2006). *Steps for better thinking: Developmental problem solving process.* Downloaded 6/20/2007 from http://www.wolcottlynch.com/EducatorResources.html