AC 2010-408: INTEGRATING COMPUTER PROGRAMMING TECHNOLOGIES INTO THE INDUSTRIAL ENGINEERING CURRICULUM

Jorge Valenzuela, Auburn University

Jorge Valenzuela received his Ph.D. in Industrial Engineering from the University of Pittsburgh in the year 2000. He is currently an Associate Professor in the Department of Industrial and Systems Engineering at Auburn University. His recent research involves stochastic models for the evaluation of production costs and optimization of electric power generation. He teaches courses on Operations Research and Information Technology.

Jeffrey Smith, Auburn University

Jeffrey S. Smith is Professor of Industrial and Systems Engineering at Auburn University. Prior to this position, he was on the faculty of the Industrial Engineering Department at Texas A&M University. In addition to his academic positions, Dr. Smith has held professional positions with Electronic Data Systems (EDS) and Philip Morris USA. Dr. Smith has a BS in Industrial Engineering from Auburn University and MS and Ph.D. degrees in Industrial Engineering from Penn State University. His primary research and teaching interests are in manufacturing systems design and analysis and discrete event simulation.

Ben Reece, Auburn University

Ben Reece obtained a Bachelor degree in Industrial and Systems Engineering from Auburn University in 2008. He is currently a Masters student in the Industrial Engineering Department at Auburn University. He is also member of IIE.

David Shannon, Auburn University

David Shannon has a Ph.D. Research Methodology and Statistics from the University of Virginia in 1990 and is currently the Humana-Germany- Sherman Distinguished Professor at Auburn University. His research has focused on assessment issues, program evaluation and methodological issues. Dr. Shannon has published over 50 articles in refereed journals since joining the Auburn faculty in 1990.

Integrating Computer Programming Technologies into the Industrial Engineering Curriculum

Abstract

Incorporation of powerful “scripting languages” in engineering modeling software is becoming increasingly common. Unfortunately, while most engineering curricula include one or two programming-related courses at the freshman and/or sophomore level, students generally show weak computer programming skills when they reach the core curriculum courses. This project seeks to develop an innovative set of classroom modules involving computer programming for use throughout the Industrial Engineering curriculum. The modules are in response to our belief that the main cause of the problem is not the specific material covered in the computer programming courses but the lack of reinforcement that the students receive from other engineering courses with regard to computer programming skills. This project’s goal is to investigate whether significant, formal, well-designed reinforcement of the programming skills outside of traditional programming courses will lead to students more proactively using their programming skills in situations that would benefit from their use. Five modules have been developed and tested during the first year of the project. In this presentation, we will discuss the preliminary results stemming from the use of these modules in our undergraduate courses. This project is being funded by the Division of Undergraduate Education of the National Science Foundation under Grant DUE-0836260.

Introduction

As educators, it is our responsibility to provide the necessary knowledge and skills for engineering students to be successful in their workplace. In this regard, we must examine the use of computer technologies in today’s business and engineering environments. Experts agree that most of the engineering modeling software used today in the industry requires some knowledge of a computer programming language such as FORTRAN, C, C++, Java, or Visual Basic (VB). Moreover, the incorporation of powerful “scripting languages” in most of these modeling tools has significantly increased the capability of integrating them and knowledge of these scripting capabilities is becoming more important for engineering students.

Most current engineering curricula include one or two courses at the freshman and/or sophomore levels that cover general computer programming using one or two computer languages. In the industrial engineering (IE) curriculum at Auburn University there are two courses dedicated to teaching the basics of computer programming. COMP 1200, which is offered at the freshman year, teaches MATLAB and COMP 1210 covers Visual Basic and it is generally taken during the sophomore or junior year. However, despite successfully completing these courses, the students generally show weak computer programming skills when they reach the core IE courses during their junior and senior years. In response, we have revised the content of these two courses on several occasions (changing them from teaching the programming language C to C++ and then to Java), but the problem persists. Last year, our department decided to again change the content of COMP 1210 from Java to VB. The new course was initially offered during the fall semester of 2008. The reason for teaching VB is that the preferable computer modeling tool used by the majority of IE instructors and practitioners is Microsoft Excel. Excel is a spreadsheet modeling tool considered to be a de facto standard. Excel and similar tools reduce the programming burden by providing large sets of pre-programmed functions. However, in many modeling situations it is necessary to create new functions or modify macros, which requires knowledge of the VB programming language. Furthermore, VB and related scripting languages/tools can be used to significantly enhance the power of many of today's engineering computer applications. In addition to Excel, the functionalities of Arena (simulation modeling), Visio (flowchart development), AutoCAD (computer-aided design) and MS Access (database management) can all be expanded through the use of VB and these applications are very commonly used by industrial engineers.

However, if we do not *reinforce* the computer programming-related learning throughout the entire curriculum, the latest change in programming language from Java to VB will probably have limited success once again. We have come to believe that the main cause of the problem is not the specific material covered in the computer programming courses but the lack of reinforcement that the students receive from other engineering courses with regard to computer programming skills. Only one or two of the courses in the IE curriculum include computer modeling assignments that explicitly involve programming. As such, students do not learn to think of their programming skills when faced with a problem that would benefit from their application. To address this problem, we have developed with the support of the NSF Division of Undergraduate Education, a set of instructive modules for some IE professional courses in our curriculum. Our long term goal is for the students to learn to *recognize* opportunities to apply their programming skills in solving engineering problems without having to be explicitly told to do so. During Phase 1 of the project, we are exploring the feasibility of our framework and have developed and evaluated five instructive modules for four IE professional courses in our curriculum.

Modeling Framework

The methods that are commonly used to achieve educational objectives include: lectures, experimental laboratory, design projects, case studies, games, and internships[1]. All of these methods are used in teaching IE courses in the Industrial and Systems Engineering Department at Auburn University. However, the lecture method is by far our most common approach for instruction. This is also the case at most other engineering academic departments. One of the drawbacks of the lecture method is that it relegates students to a passive role in the classroom. One approach to overcome this deficiency and engage students in a more active role is to add case studies to the teaching[2]. In industrial engineering, a case is usually a description of an actual situation that commonly involves a problem and/or a decision faced by a person or group of people in an organization. As engineers, we are constantly confronted to solve problems. We usually have a large set of data and use a "problem solving technique" to find a "solution" to the problem. In order to solve the problem, we are likely to apply the following steps: understand the problem, analyze its causes, identify alternatives, formulate a model to assess the alternatives, select one alternative, implement it, and evaluate the results in order to determine whether or not the problem has been solved. It is interesting to note that students know these steps by heart but usually have only a vague understanding on how to *apply* the steps in formal or well-defined way. Case studies that involve problem solving skills are therefore of great value to the students as they provide practical experience in real problem solving.

In our framework, we use the case studies approach as the pillar to fully integrate computer programming in the IE curriculum. Figure 1 illustrates our framework. The framework includes three levels. The first level corresponds to the development of two comprehensive case studies. We focus the cases on two fundamental areas of industrial engineering – manufacturing and services. For the manufacturing case, we will concentrate on the automotive manufacturing industry. The reason for the choice of this industry is that due to business incentives, rural development, and low-cost and plentiful labor available in the southeastern United States, many automakers and their corresponding tier 1, 2, and 3 suppliers have recently established manufacturing complexes southeastern states. In Alabama, these include industry leaders such as Mercedes Benz, Toyota, Honda, Hyundai, and KIA. As these facilities move into the southeast, the automotive industry is fast becoming one of the primary employers of our graduates. For the services industry case, our focus will be on health care organizations such as hospitals, clinics, and managed care providers. The reason for this choice is that health care is the fastest growing service industry in the US. Currently, there is a great demand for professionals that can analyze, manage, and operate healthcare systems and industrial engineers can supply that need. As the baby-boom generation ages, this growth is expected to continue for the foreseeable future. The second level in the proposed framework includes the instructive modules. The third level specifies all of the professional courses from the IE curriculum that will use the modules.

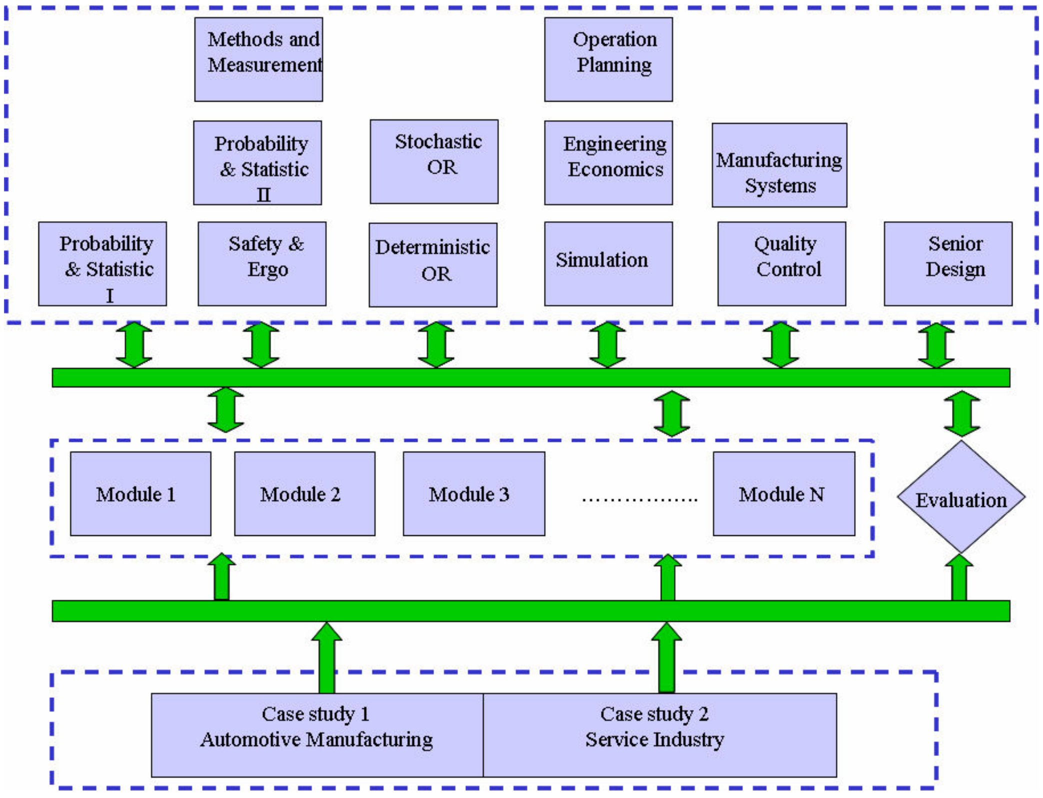

Figure 1: Case study-based framework for integrating computer programming

Module Design

In this exploratory phase of the project, we have designed modules that deal with the components of an automotive manufacturing system. We have categorized the modules along two dimensions: the *level of guidance* that they require; and the *level of difficulty* that they involve. Thus, a module is denoted by two superscript letters. Figure 2 illustrates the model in detail. For example, a $G^L D^L$ module consists of a problem of low guidance (level L) and low difficulty (level L). We set three values (High, Medium, and Low) for both guidance and difficulty levels. In a G^H module, the student is asked to program a particular task using a specific computer programming language (a high level of guidance). These types of modules are be used to assess the retention of the concepts learned in COMP 1200 and COMP 1210. In a G^M module, the student is asked to solve a problem using a computer tool. In this case, the student has to determine the most appropriate computer programming language for the problem since the guidance level will be less. Similarly, a G^L module consists of open-ended problems, which may have many correct answers and several different ways to be solved and the student must make many design and implementation decisions. It is our hope that the students will recognize the potential applicability of their programming skills for these modules. As such, G^L and G^M type modules are used to measure the increase students' cognizance of and proactiveness in using computer modeling skills. The level of difficulty of a module is related to the level of effort required to complete the assignment. Faculty feedback is used to set and control these values. As we have just completed the initial module offering, we will be reviewing these values.

		Guidance Level			
		High	*Medium*	*Low*	
Difficulty	*High*	$G^H D^H$	$G^M D^H$	$G^L D^H$	D^H
	Medium	$G^H D^M$	$G^M D^M$	$G^L D^M$	D^M
	Low	$G^H D^L$	$G^M D^L$	$G^L D^L$	D^L
		G^H	G^M	G^L	

Figure 2: Module type description model

Module Structure

A module consists of student-related material and instructor-related material. The student material contains the module objectives, problem description, requirements, and any required data sets. The instructor material includes the problem solution, a handout describing the technical subjects involved in the assignment, and a survey to provide feedback on the module. This survey is used to measure the success of engaging faculty members in the process. The surveys are also used to adjust the guidance and difficulty levels of the modules for subsequent offerings. It is important to mention that not all people learn in the same way. We learn by listening, seeing, and doing, one of which is more dominant in each person. In fact, each individual learns by using a complex combination of these three elements[3]. Therefore, we have included in some of the modules audio-visual material. To illustrate our approach, two developed modules are described below. The first module requires the use of the Matlab programming language, while the second one requires the Visual Basic for Applications (VBA) for Excel programming language.

Module Example 1

Module Title: Making Decisions using Markov-chain theory

Guidance Level: High   **Difficulty level:** Low

Course: Stochastic Operations Research**,** INSY 3400

Module Aim: The aim of this module is to increase students' cognizance of and proactiveness in using computer modeling skills.

Module Objectives: Upon completion of this module, students should be able to

1. Design an algorithm to compute the steady state probability of a Markov Chain
2. Code the algorithm using Matlab programming language
3. Use the algorithm to evaluate economic decisions

Problem Description: Tiger Motor Company's Auburn facility produces several different vehicle types such as two and four door sedans. The plant uses many different robotic weld guns (see Figure 3) to assemble all body types produced. The robotic weld guns have a current of 12 to 14,000 amperes per weld and can make 20 or more welds per minute. The temperatures on these robotic welders reach anywhere from 150 degrees fahrenheit to over 500 degrees fahrenheit. Given this temperature range, it is extremely important to maintain proper cooling systems. With the large amount of welds performed and the resulting high temperature conditions, the robotic weld guns will produce defective welds and can fail frequently. Tiger Motor Company must replace the entire gun if it fails at a cost of $7,000 excluding downtime and labor costs. It takes 1 ½ hours to replace the gun, and the maintenance worker installing the new gun earns $60 an hour. The facility produces 20 vehicles per hour with a profit of $1,000 per vehicle. Tiger Motor Company considers the downtime cost as the potential profit lost when vehicles cannot be produced. The Tiger Motor Company is deciding which tool replacement policy to use in order to reduce costs. The policies determine how long the replacement cycle is and are based on monthly replacements. The robotic weld guns have a 1/12 chance of breaking down during the first month of operation and this probability increases linearly by 1/12 each month until 12 months. After 11 months, if the robotic weld gun has not failed it is replaced by a new weld gun. The robtic weld gun has a monthly maintence cost of $4,000 after this first month of operation and increases by $2,000 each month over the lifetime of the machine. The company wants to evaluate the cost of the replacement policies from 2 to 10 months and find the replacement policy with the lowest expected cost.

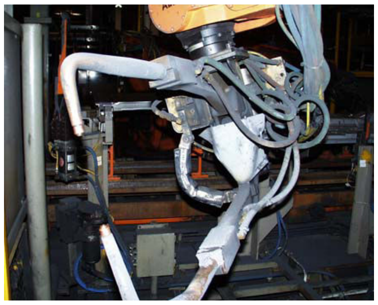
Figure 3: Robotic Welding Gun

Requirements: The managers at Tiger Motor Company have defined the Markov chain to be the age of the machine at the beginning of the month, i.e. 0, 1, 2, … They need a computer program to assess each replacement policy. You are asked to:

1. Design and code an algorithm in Matlab that is able to:
   a) Input the number of months in the replacement cycle or policy
   b) Generate a transition probability matrix for the given number months. As an example, Figure 4 shows the transition probability matrix for the 3-month replacement police while Figure 5 shows the matrix for the 6-month policy
   c) Store these values in a matrix called "ProbMatrix"
   d) Calculate the steady state probabilities for this Markov chain by raising "ProbMatrix" to the power of 100 and store the result in a vector called "SteadyState"
   e) Compute the expected cost for this policy using the giving cost information

2. Use the code to evaluate the expected cost for the policies ranging from 2 to 10 months and find the policy with the smallest expected cost

	0 (new)	1	2	3
0 (new)	1/12	11/12	0	0
1	2/12	0	10/12	0
2	3/12	0	0	9/12
3	1/12	11/12	0	0

Figure 4: Transition Probability Matrix (P) for 3-month Policy

	0 (new)	1	2	3	4	5	6
0 (new)	1/12	11/12	0	0	0	0	0
1	2/12	0	10/12	0	0	0	0
2	3/12	0	0	9/12	0	0	0
3	4/12	0	0	0	8/12	0	0
4	5/12	0	0	0	0	7/12	0
5	6/12	0	0	0	0	0	6/12
6	1/12	11/12	0	0	0	0	0

Figure 5: Transition Probability Matrix (P) for 6-month Policy

Module Example 2

Module Title: Creating Control Charts

Guidance Level: Medium **Difficulty Level:** High

Course: Statistical Quality Control, INSY 4330

Module Aim: The aim of this module is to increase students' cognizance of and proactiveness in using computer modeling skills.

Module Objectives: Upon completion of this module, students should be able to

1. Design an algorithm to group sampled data into subgroups and detect when the process is out-of-control
2. Plot the sub-grouped samples onto a control chart
3. Code the algorithm using Visual Basic for Excel programming language

Problem Description: Tiger Motor Company is always on a mission to meet superior quality vehicle standards. One component of their vehicles that Tiger Motor Company is especially proud of is the performance and lifetime of their engines. The most popular engine assembly used at Tiger Motor Company is the 2.3 liter engine which is used in their SE 500 sedan and their K-300 small truck. Piston rings are an essential contributor to ensure an extensive engine lifetime, so the quality control department has chosen to monitor the variance of the 2.3 L engine piston ring widths. Pistons have rings that wrap tightly around the piston to assure no leakage of the high temperature gases produced during power stroke in the combustion chamber. If the piston rings do not fit well enough, it is possible that the engine will consume more oil than is necessary, and on an extreme level it could cause an engine fire. The piston rings also help keep the pistons from overheating when the combustible mixture explodes to power the vehicle. The piston rings help maintain lubrication between the pistons and the cylinder walls so that there is less wear and tear between them. The performance of the piston rings can greatly affect the overall performance, lifetime, and safety of the entire engine during its lifetime.

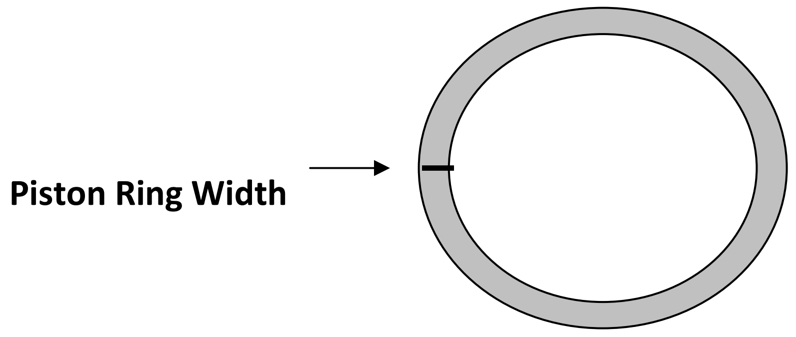

Figure 6: Piston and piston rings

The piston rings used in the 2.3 liter engine assembly are being scanned by a computer vision system that computes (measures) and records the piston ring widths every 1 second. After a phase 1 statistical analysis, the Quality Department of Tiger Motor Company has determined that when the process is in-control the ring widths are normally distributed with mean 0.0775 inches and standard deviation 0.000373 inches. The quality control management needs a computer program that will be implemented in their engine assembly facility to monitor (second phase) the performance of the piston ring molding process. They want the program to take width measurements of the piston rings and generate an Xbar chart to display the process performance. The computer program will assist management in the engine component quality department to monitor the performance of molding processes and help prevent rings that are the wrong size from being installed on engines. This prevention will help decrease engine failures and prevent the company from spending money on replacing engines and pistons.

Requirements: The company managers need a computer program to generate a control chart for this process. The computer vision data will be generated with a function that is provided to you in Figure 7.

```
Public Function Sample_from_Computer_Vision() As Double
   Mean = 0.0775
   StdDev = 0.000373
   Ring_Width_Sample = Application.WorksheetFunction.NormInv( Rnd( ), Mean, StdDev )
   Application.Wait ( Now + TimeValue("0:0:1") )
   Sample_from_Computer_Vision = Ring_Width_Sample
End Function
```

Figure 7: Computer Vision Function

This function will generate random piston ring width measurements once every second, simulating the sampling process in real time. You are asked to:

1. Compute the Upper Control Limit (UCL) and the Lower Control Limit (LCL) for a sample size of 5 observations (use equation 6.14 from the textbook)
2. Create an empty chart in Excel to plot the data. The chart should also include lines on the graph for the centerline, UCL, and LCL
3. Code an algorithm to sample the pistons' widths and group them into a sample of 5 observations. Compute and plot the averages of each group of 5 observations onto the Xbar control chart. The chart should be updated every 5 seconds to include the new sampled measurements and should show just the last 20 groups
4. Add a code to detect whether the process is or is not in-control according to the Western Electric Handbook's rules for detecting nonrandom patterns on control charts. Indicate with a message when the process is out-of-control giving the rule number that was broken and stop monitoring the process; otherwise, stop the program after 200 seconds. The rules for an out-of-control process are given below
5. Emulate a shift of the mean of the process by changing in the computer vision function the value of the mean to 0.0779 and run the program again. How many seconds does it take to detect that the process is out-of-control?

Module Evaluation and Findings

The evaluation of this exploratory phase of the project is based on the performance of the students and the experiences of the faculty members that used the developed modules[4]. Before and during data collection, we adhered to the human subject-related procedures and policies of Auburn University. Data were collected in terms of surveys and participant's academic performance. Participants were clearly informed the purpose of the data collected to how the results would be used. While all students were required to complete the modules as part of the respective courses, students were given the option for their scores to not be used in the analysis of the modules. To date, no students have made this request. Table 1 gives information about the modules that were developed and used during the initial offering (fall semester of 2009).

Table 1: Modules developed and offered during fall semester of 2009

Module	Code	Course
Making Decisions using Markov Chain Theory (Module Example 1, above)	$G^{H}D^{L}$	INSY 3400 – Stochastic Operations Research
Queuing Analysis	$G^{M}D^{M}$	INSY 3400 – Stochastic Operations Research
Coding the Simplex Method	$G^{M}D^{M}$	INSY 3410 – Deterministic Operations Research
Control Charts (Module Example 2, above)	$G^{M}D^{H}$	INSY 4330 – Quality Control
Assembly Line Balancing	$G^{L}D^{H}$	INSY 4700 – Manufacturing Systems

The modules were developed with the assistance of the faculty members responsible for teaching these courses. The assignment, grading, and student advising associated with the modules were all done by the project investigators and graduate students. The full content of these modules are available at the website developed for this project http://sim.auburn.edu/ccli. Videos were also taped to provide a better understanding of the requirement of the modules. They are also available at the project website.

Students were given two weeks to complete each module. After each module was due, an in-class announcement requesting that students complete a feedback survey was made. The feedback survey included course and module information, attitudes toward the class, and module feedback. More specifically, students were asked to identify the module they just completed, the course of enrollment and whether they had previously completed any instructional modules in their coursework. The remaining items were used to form four measurement scales.

There were 19 items pertaining to student attitudes toward the class. Of these 19, eight were aimed at measuring students' self-efficacy, six regarded the value of the course, and 5 sought information about the extent to which students were required to think critically in the class. Finally, 14 items were included to gain feedback about each module in terms of guidance provided to students to complete the module, fit with course, and benefits of the module.

Reliability estimates were very supportive, ranging from 0.806 for critical thinking to 0.923 for course value.

Feedback pertaining to each module was gathered through 14 items, using a five-point Likert-scale. Overall, student feedback has been moderately supportive. The average feedback response was below the scale midpoint of "3" for three of the five modules, with averages ranging from 2.25 to 4.25. This information is summarized for each module in Table 2.

Table 2: Module Feedback

Module	Survey Responses	Feedback - Mean (SD)
Creating control charts	36	2.33 (0.70)
Coding the LP simplex algorithm	16	3.13 (0.92)
Markov-chain theory application	14	2.92 (0.89)
Balancing the assembly line with COMSOAL heuristic	26	2.25 (0.85)
Studying the effects of system parameters of M/M/s queuing model	2	4.25 (0.86)

More detailed information is provided by student responses to each of the 14 items. These 14 items are presented below:

1. I understood what was expected of me in completing this module.
2. I was NOT prepared with the skills necessary to complete the module successfully.
3. The directions were easy to follow.
4. The module demonstration was very helpful.
5. There was NOT enough information provided to successfully complete the module.
6. This module helped me develop my computer programming skills.
7. I will be able to use what I learned from this module in future courses or a job.
8. This module did NOT seem to fit with the rest of the class content.
9. I enjoyed working on this module.
10. More modules like this should be integrated in existing courses.
11. This module difficulty was appropriate for the class.
12. I struggled to complete this module successfully.
13. This module made effective use of audio and visual features.
14. I look forward to completing more modules like this in other courses.

The average response to each item for each module is summarized in Table 3.

Table 3: Survey summary of questions regarding the modules

	Module1	Module2	Module3	Module4	Module5
	(n=36)	(n=16)	(n=14)	(n=26)	(n=2)
Item	Mean (SD)	Mean (SD)	Mean (SD)	Mean (SD)	Mean (SD)
1	3.47 (1.06)	3.60 (1.40)	3.64 (1.08)	2.85 (1.41)	4.50 (0.71)
2	1.78 (1.07)	2.94 (1.43)	3.07 (1.32)	2.12 (1.51)	5.00 (0)
3	3.17 (0.91)	3.31 (1.14)	2.86 (0.95)	2.38 (1.36)	4.50 (0.71)
4	2.56 (1.16)	3.44 (1.37)	3.07 (1.33)	1.52 (0.96)	4.50 (0.71)
5	2.75 (1.11)	3.69 (1.01)	3.36 (1.22)	2.35 (1.44)	4.50 (0.71)
6	2.56 (1.25)	3.00 (1.27)	3.36 (1.22)	2.77 (1.48)	4.50 (0.71)
7	2.25 (1.08)	2.75 (1.29)	2.93 (1.44)	2.60 (1.56)	4.50 (0.71)
8	2.94 (1.26)	3.88 (1.48)	3.36 (1.15)	2.81 (1.55)	4.50 (0.71)
9	1.61 (1.02)	2.81 (1.33)	2.62 (1.50)	2.00 (1.33)	4.00 (1.41)
10	2.00 (1.14)	2.69 (1.45)	2.36 (1.34)	2.08 (1.41)	4.00 (1.41)
11	2.11 (1.06)	3.25 (1.24)	3.00 (1.04)	2.00 (1.33)	4.50 (0.71)
12	1.92 (1.18)	3.13 (1.26)	2.21 (1.25)	2.69 (1.64)	3.50 (2.12)
13	2.11 (0.95)	2.94 (1.24)	2.71 (1.14)	1.69 (0.97)	3.00 (0)
14	1.47 (0.77)	2.44 (1.41)	2.43 (1.56)	1.62 (1.02)	4.00 (1.41)

Further examination of the responses to specific feedback items reveals some findings. These initial findings do not include information for module 5 because there are just 2 responses to date.

Overall, the responses are most favorable for modules 2 and 3. For module 2, 8 items (out of 14) received an average response at or above the scale midpoint while 7 items received this level of response for module 3. It is important, however, to keep in mind that feedback for these modules is based on less than one-third of the enrolled students. While modules 1 and 4 received feedback from a greater percentage of enrolled students, the average responses were lower, with just two items receiving a response at or above the midpoint for module 1 and no items for module 4.

Across all four modules, students understood what was expected of them as this item (item 1) received one of the top 3 highest rated items for each module. Students also supported the validity of each module, including item 8 in their top 5 across all four modules. Other items receiving generally favorable responses included items 3, 5, and 6. That is, students generally believed that the directions were easy to follow, there was enough information to complete the module, and the modules helped develop their computer programming skills. On the other hand, students did not response favorably to items 9 and 14. That is, they didn't enjoy working on modules and don't look forward to completing more of them in other classes.

Conclusions

This paper describes an NSF-sponsored project involving the development of programming modules for professional classes in the undergraduate industrial engineering curriculum. The goal of the project is to investigate whether significant, formal, well-designed reinforcement of the programming skills through these modules and outside of traditional programming courses will lead to students more proactively using their programming skills in situations that would benefit from their use.

While we are very early in the project and have data from only five modules, the results are interesting and we are pleased with the progress of the project and look forward to continuing the work over the next few semesters. In general, the modules were well-received by the faculty members and, to a certain extent, the students. We will use some of the module feedback to improve subsequent modules, where appropriate and we expect to be able to draw more significant and meaningful conclusions once we have more data from more modules and from multiple replications of the same modules.

Bibliography

1. Parsons, J. and P. Klukken, "An Introductory Design and Innovation Course at the University of Tennessee," 1995 Frontiers in Education Conference, 1995, pp. 3a5.13-3a5.15.
2. Teaching and Learning with Technology, Using Cases in Teaching, available at http://tlt.its.psu.edu/suggestions/cases/index.html, Nov 11, 2009.
3. The Harriet W. Sheridan Center for Teaching and Learning, Constructing a Syllabus: a handbook for faculty, teaching assistants and teaching fellows, third edition, Bown University, 2006.
4. Westat, J. F., NSF, “The 2002 User-Friendly Handbook for Project Evaluation,” available at http://www.nsf.gov/publications/pub_summ.jsp?ods_key=nsf02057, Nov 15, 2009.